SOME EFFICIENT REPRESENTATIONS OF 3D INFORMATION BASED ON OCTREES

A THESIS SUBMITTED BY

SOJAN LAL P.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

UNDER THE FACULTY OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI – 682 022, INDIA

2000

29th December 2000

CERTIFICATE

This is to certify that the thesis entitled **"SOME EFFICIENT REPRESENTATIONS OF 3D INFORMATION BASED ON OCTREES"**, is a report of the original work carried out by **Mr. P. Sojan Lal** under our supervision in the Department of Computer Science, Cochin University of Science and Technology. The results enclosed in this thesis or part of it have not been presented for any other degree.

[illegible]

Dr. A. Unnikrishnan
(Supervisor)
Scientist "F",
Division Head (I.P.),
NPOL,
Cochin – 682 021

[illegible]

Dr. K. Poulose Jacob
(Supervisor)
Professor and Head,
Department of Computer Science,
Cochin University of Science
and Technology, Cochin – 682 022

Contents

Contents

Contents

List of illustrations

Chapter 1
Overview of 3D Representation

Introduction

Computer Graphics is one of the most exciting fields in Computer Science, in view of its capability to present results in a most palatable visually perceivable form. It is a complex and diversified technology, which can be subdivided into manageable parts by considering the end product for better understanding. The picture may be an engineering drawing, design of a product to be manufactured, an exploded parts illustration (for a service manual, presentation of a product display in exhibition/ seminars), a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, a single frame from an animated movie, or a simulation system. The picture is the fundamental cohesive concept in computer graphics [1Rogers88]. Recently interactive computer graphics, which gives two-way communication between computer and user, has gained popularity. Flight simulators are a mock up of an aircraft flight deck, containing all the usual controls and surrounded by screens on which are projected computer generated views of the terrain visible on takeoff and landing. CAD/CAM Engineers, molecular biologists, town planners, transportation engineers and trainers are some of the few end users of interactive computer graphics [2Newman97].

While the instances cited above largely describe 2D images, applications of computer graphics in medicine address the issues of representing 3D information. Apart from standard X-ray images extensively used in medicine, surgical planning using computer constructed images and computer assistance to actual surgery are now common in both orthopedic and neuro-surgical procedure [3Rhodes90]. Computer Tomography (CT), Magnetic Resonance (MR), Positron Emission Tomography (PET) [4Derbin88] [5Porter84] and more recently ultrasound have come up as medical imaging modalities that deliver cross sectional images of the human body with the depth information, which is not available in conventional X-ray image [6Tiede90]. Adjacent CT/MR images recorded in sequence represent a 3D volume of that part of anatomy scanned [7Brewster84]. The volumetric information on human body gives the idea of the size of the organs, tumour and their positional relationships. In addition to displaying the physical structure, 3D imaging has been particularly used to displaying radiation beams and isodose surface for radiation therapy planning [8Farrell85] [9Maguire91].

An X-ray source, typically a heated tungsten filament, releases electrons that accelerated by a high electric field, strike a tungsten target. The energy of the impact dislodges electrons from the inner electronic shells, and the consequent filling of the "holes" causes the release of X-ray radiation. The energy of this electromagnetic radiation is such that it passes through exposed portions of the human body. In most diagnostic radiology, a simple X-ray "shadow" of the body part of interest is projected onto a film (or electronic amplifying medium). The ability to distinguish different structures in such a projection shadowgram is limited in several ways. First, to clearly distinguish two adjacent regions, their transmittance must be significantly different. Thus on "Plain radiographs", bones, which absorb much more X-rays than other tissues, and normal and abnormal structures in the lungs which are surrounded by air, can easily be recognized and studied. However, regions with similar X-ray transmittance, such as most of the "soft tissue" (muscle, organs, blood etc.) of the body, cannot be distinguished from one another in conventional X-ray images. A second major limitation in projection X-rays is the superimposition of images of overlaying structures. This further decreases the ability to identify normal and abnormal structures.

X-rays generates Computerized Tomographic images called CT or CAT scans. A fan of X-ray beams passes through a slice of the body and is recorded on a transducer array as a line shadow or projection. The X-ray source rotates around the body, producing hundreds of these projections of the slice from many different angles. A computer then uses all the projections from a single slice to compute, by estimation or back calculation, the internal structures in the slice that must have produced those projections. The end result is an image of a cross section of the body. We can process CT images very easily as it is inherently a digital image.

Computerized Tomography (CT) is a medical imaging technique that has revolutionized the field of medical diagnosis to the extent of pinpointing malignancies and hemotoma. The term tomography (tomo derived from Greek word for slices) refers to cross sectional imaging of objects [10Lovey90]. The most spectacular success of tomography is in diagnostic imaging with non-diffracting sources such as X-rays. (However, there has been an increasing interest in extending this image formation technique to diffracting sources; eg. using ultrasound [11Greenleaf83], and Nuclear Magnetic Resonance (NMR) [12Hinshow83]). The result was the production of high quality images, as a reliable diagnostic support.

The various absorption values of CT scan ranges from -1024 for air, to +1024 for bones. The images are reconstructed during the scan. The image captures on a SIRETOM 2000 [13Siemens] are reconstructed during CT scan. This range of absorption values facilitates to filter different organs, either to display the organs of our interest or to remove it from the display.

Like CT scans, Magnetic Resonance (MR) scans produce cross-sectional images, but the tissues they show look very different because this kind of imaging uses a different physical principle. A strong field, about 1 tesla, or 1000 times the strength of the earth's magnetic field, rotates polar atoms in the body (mostly hydrogen) into alignment with the field. These atoms are not static; they spin and process, and the axis of spin tends to align with the external magnetic filed. An external pulse of radio frequency (RF) energy excites and disturbs these spinning atoms, and, as each atom relaxes back to its original spin axis, it emits an RF signal. This allows the imaging equipment to detect the emitter's location [14Dev99].

The frequency of the spin is proportional to the applied magnetic field. By using additional magnetic fields, we can control the spin at different parts of the body, thus allowing different locations. Further the hydrogen atom has different mobility in different tissues, which allows differentiation among the tissues in the imaging.

The different imaging physics of the MR and CT techniques make them appropriate in different contexts. MR images are excellent for displaying differences in hydrogen-rich tissues (soft tissues); CT images are better for bone details and tissues filled with radio-opaque contrast material. As an example we can visualize the detail of gray and white matter in the brain very clearly in an MR image, but we cannot see it at all in a CT image.

Ultra sound imaging uses the reflective and refractive interactions of sound with tissue to image different tissues. Within a selected range of acoustic frequencies, about 1 to 20 Mhz, sound waves can penetrate human tissue. An array of piezoelectric crystals emits a fan beam of ultrasound, a pressure wave that moves out through the tissue. At boundaries, the tissue's refractive properties change, reflecting a portion of the sound wave back to the receiver. The time interval between emission and reception provides a measure of the distance of the reflecting surface, and the imaging equipment uses this measurement to create the ultrasound image.

The change in frequency of reflected sound when it encounters moving tissue such as blood allows ultrasound imaging to move beyond anatomy into the imaging of function. Ultrasound is particularly appropriate for imaging tissues just below the body surface. When imaging deeper structures, the physician uses techniques to reduce the number of reflective surfaces between the transducer and the structure. As an example, to image the foetus in the uterus the patient drinks a large quantity of water so that the bladder fills and swells to a position between the body surface and the uterus. The uniform refractive characteristics of water allow the sound to penetrate to deeper tissue.

Nuclear medicine imaging includes a vast range of methods, all based on the injection into the body of radio-pharmaceuticals (radiation producing chemicals) that different organs take up selectively. We use the radiation emitted by these chemicals to image the body in a manner similar to X-ray radiography. For example, the thyroid takes up iodine selectively. We use radiation-producing iodine123 to image the thyroid to detect enlargement.

The amount of uptake also has clinical importance. Metabolically active regions, such as tumours or inflated tissue, may show increased uptake, whereas tissues with reduced blood flow, such as heart tissue after a heart attack, may show decreased uptake. On this principle, we can use thallium-201 to image heart muscle that may be inadequately perfused with blood. We can use gallium-67, technetium-99, or thallium-201 to image tumours; technetium-99 to image bone; xenon-133 to image lung perfusion by air; and fluorine-18 to detect regions of increased metabolism.

In medicine, a number of technologies produce volumetric data sets. Potentially they are of any dimension, but typically three dimensional (four dimensional in case of time varying data). These techniques to produce volumetric data include Magnetic Resonance (MR), X-ray computed tomography (CT), positron emission tomography (PET) and more recently ultrasound. These data sets range in size from a few megabytes in case of MR studies of small organs to tens of giga-bytes of the Visible Human data sets [15Senger99].

Three-dimensional imaging is designed to facilitate the interface between the radiologist and clinician. The data acquired from a transaxial CT scan is usually displayed in a 2D plane format. Although this display technique is generally quite adequate for diagnosis, it does not optimally communicate the 3D nature of the involved anatomy or the full extent of pathology; 3D imaging presents complex anatomical findings in a format easily understood without prior specialized training and experience.

Computer Assisted Surgery [16Ludwing90] using electro-mechanical 3D coordinate digitizer, having 6 joints for the positioning of stylus from the measured angles and the given arm length, which is communicated to the host computer via RS-232 link are used for 3D co-ordinate measurement. The combination of 3D co-ordinate measurement technique, voxel processing methods, and pseudo 3D image presentation supports pre-operative planning therapy, path finding during operation and postoperative therapy control. The co-ordinate measured by the electro-mechanical digitizer is co-related with a voxel model of the object gained by a preceding CT examination.

Surgical planning [6Ulf-90] [17Linda84] using computer constructed images and computer assistance to actual surgery are now common in both orthopedic and neuro-surgical procedure [18Michael90]. CT and MRI have come up as medical imaging modalities that deliver cross sectional images of the human body with the depth information, which is not available in conventional X-ray images. Adjacent CT images recorded in sequence represent a 3D volume of that part of anatomy scanned. The volumetric information on human body gives the idea of the size of the organs, tumour and their positional relationships. In addition to display the physical structure, 3D imaging has been particularly used to display radiation beams and isodose surface for radiation therapy planning [10Lovey90] [8Edward85].

One of the classical problems in model-based vision is the estimation of the pose (i.e. location and orientation) of a 3D object with respect to a scene described by a system data. In Computer and Robot Assisted Surgery, it is necessary to match 3D medical images (MRI or CT) with 2D X-ray projections [19Taylor95]. Prior to a surgical operation, a simulated intervention procedure is performed on 3D images (MRI or CT) of the patient. Then, at the beginning of the operation, X-ray images of the patient lying on the operating table are taken from two viewpoints. The geometry of the X-ray projections is related to a robot's reference frame using standard calibration techniques [20Champleboux92]. To accurately perform robot assisted surgery based on the pre-operative simulation, the exact position of the patient must be determined by matching the reference system Ref3D associated with the pre-operative images with the reference system $\text{Ref}_{\text{sensor}}$ associated with the X-ray images taken in the operating room [21Sautot92]. In model based vision, this problem can be formulated as the estimation of the special pose of a 3D smooth object from 2D video images. [22Lavallee and Szeliski 95] present a method for determining the rigid body transformation that describes this match. To quickly and accurately compute distances to the surface, Lavallee and Szeliski introduced a pre-computed distance map represented using an octree spline whose resolution increases near the surface. This octree structure allows to find the minimum distance along each line, using breadth-first search.

Applications of observation based modeling include, among others: creating models for animation, reconstructing human body parts for surgical planning, recovering machine parts for virtual factory simulation, and building CAD models for model-based recognition. Some observation based modeling techniques involve motion estimation between successive pairs of views in a sequential manner [23Parvin92] [24Turk, Levoy94] [25Higuchi95].

Computer Aided Design (CAD) systems were essential for 2D drawing and drafting. Solid modeling techniques emerged to describe 3D products unambiguously [26Requicha92] and have seen a proliferation of solid modelers and 3D CAD systems [27Han98]. In industry, engineers extensively use both CAD and Computer Aided Manufacturing (CAM) techniques to design and manufacture products. However, very little communication occurs between CAD and CAM [28Bedworth91]. Computer Aided Process Planning (CAPP) emerged as the communication agent between CAD and CAM. CAD data consists mainly of geometric information about the product. Given CAD data, CAPP generates a sequenced set of instructions to manufacture the specified product.

Papers published in 1980s for the visualization of objects from gray-level volumes include [29Goldwasser85] [30Lenz86] [31Hoehne86] [32,33Hoehne87] [34Smith87] [35Loransen87] [137Levoy88] [36Drebin88] [37Hoehne89].

1.1 Representation of 3D information

Representing 3D information as a three dimensional array is the most natural and straightforward extension of the 2D image. The algorithm relating to segmentation [38Cline90] [15Senger99] [39Summers97], connected component labeling [42Unni87] and fractal compression [168Waye96] have been reported in literature. The relative ease in accessing the individual voxel element in the 3D array, makes the extension of 2D algorithms for image processing straightforward. However the volume representation using 3D arrays consumes a lot of space, while representing the standard 3D images in medical imaging constructed out of CAT scan and MRI. It has been observed that the 3D images represent a volume distribution having concentration of densities in specific regions. In other words, the density information can be lumped into solid space. In the reconstructed image of brain tissue, air and bone remains segmented very clearly. This property suggests the representation of 3D information by looking for homogeneous density distribution within a given volume space. The resulting representation called octree encoding is followed through out the thesis and will be discussed in Sections to follow.

1.2 Octree Encoding

The striking feature in handling 3D information at voxel level is the volumetric nature. While the volumetric nature of 3D data is astounding, it is encouraging to note that there is often a good amount of homogeneity inside the solid profile. The homogeneous property can be lumped into solid masses, the disjoint union of which can also result in the solid profile. The idea of sub dividing a solid space stems out of an earlier work in a different field by Klinger and Dayer [40Klinger76] [41UnniA-88] to represent two dimensional image as a rooted tree called quadtree. The leaves of the tree represent square tessellation of homogeneous, disjoint unions of image. Large homogeneity results in lesser number of nodes of the tree and hence saving the storage space [42UnniA-87]. The image space is usually taken to span over $2^n \times 2^n$ pixels, where n is an integer, called resolution factor.

The quadtree representation of a binary (0, 1) image array is based on recursive subdivision of the array into quadrants, until the content of each quadrant is either empty (0), or full (1).

Octrees are the 3D extensions of quadtrees, and we use octants instead of quadrants. Extensions of the quadtree into three dimensional solid representation by the use of octternary trees (octrees) has been proposed by Jackins and Tanimoto [43Jackins80], Hunter [44Hunder78], Readdy and Rubin [45Reddy78]. Following Klinger and Dayer [40Klinger76], the universal cube of volume $2^n x2^n x 2^n$ is subdivided into eight disjoint sub cubes of sizes $2^{n-1} x 2^{n-1} x 2^{n-1}$ each, using planes parallel to the sides of the universal cubes, simultaneously checking if the solid mass of same density is totally contained inside a sub-cube or not. Sub cubes totally include (or exclude) the solid mass represent a homogeneously dense portion of the solid mass, and are left as it is. However those sub-cubes that partially include the solid mass are further subdivided into 8 cubes of size $2^{n-2} x 2^{n-2} x 2^{n-2}$. The sub division continues until the smallest cube of size 1x1x1 representing the solid mass is reached. The size of the smallest cube is decided by the requirement of accuracy.

The octree data structure is basically an approximation of a three-dimensional object as a disjoint union of cubes of varying size organized hierarchically according to their size [46Chen88] [47Hunder79] [48Meagher82] [UnniB49], [Mercy50], [Yamaguchi51]. This octree encoding provides an efficient object representation of 3D voxels, for the storage of planar objects, both in image space and object space. Since pointer based octree structure requires much more storage space, there has been a considerable amount of interest in pointer-less tree structure in 1980s [52Gargantini82]. The linear octree, which stores only the location of each black node in a sorted array, has been shown to be effective in memory saving and practical in many applications [53Atkinson, Gargantini, Walsh86] [52Gargantini82] [54Gargantini86a] [55Samet and Tamminen85] [56Shafer and Samet87].

Octree encoding has recently become more popular in three-dimensional applications [57Georges98] [58Djaffer95] such as medical imaging [59Dirk98] [60Sojan98], surface rendering [61Pulli97] [62Jane92], z-buffer rendering [63Ned94], Image Contouring [64Stephane95], Ray Tracing [65Whang95], computational simulation [66Raghu98] and Occlusion culling [67Oded99]. The advantage of the octree data structure is its simple and efficient set operations [68Navazo86] (union, intersection, difference calculation, adjacency and membership testing) and transformations such as translation, scaling, rotation, etc. [69Silver97] [70Samet89] [52Gargantini82] [71Gargantini86b] on solids with octree hierarchical data structure by visiting the node utmost once. The octree can also be used for memory optimization and object oriented parallel computing. In addition to the actual voxels, global properties such as the centroid, integrated content, mass moments, volume, and circulation are computed and stored for each separate feature [72Silver91] [73Post95].

An algorithm for analyzing position uncertainties of two parts in an assembly is proposed [74Inui95]. This algorithm is applicable to 2D polygon models of machine parts, for example sections of polyhedron parts in an assembly. Since the allowable geometric variations of individual parts are relatively small in comparison with their nominal geometry, variations of the configuration are expected to be small also. Based on this assumption, "Bonding polygon" and "bonded polygon" based algorithm is developed for computing the variation bound in the octree representation.

The octree encoding provides an efficient object representation for the storage of 3D objects, both in image space and object space with a definite advantage of space saving. The universal cube of size 2^nx2^nx2^n is subdivided into eight disjoint sub cubes of size 2^{n-1}x2^{n-1}x2^{n-1} and thus each sub cube can be independently processed, either concurrently or in parallel, with increased speed of execution. This is typically realized using the concurrency feature of VAX-VMS system [75Sojan97] and parallelism by Transputer [60Sojan98] with *n* "Worker" nodes. Algorithms are available for reconstructing the solid space as it is or as sectional elevations to support Computer Assisted Surgery (CAS) and surgical planning [76Yau84].

An advantage of octree encoding is that simple operation (union, intersection, and difference of object; [77Samet87] translation, rotation, and scaling; interference detection and hidden surface removal can be accomplished by accessing one node of the tree once at the most. These operations require simple arithmetic such as integer addition, shifts, and comparisons in the context of octrees. Another useful feature of the octree approach is the ability to trade off computation time against precision. Hierarchical data structures like octrees are therefore becoming increasingly important representation techniques in the domains of computer graphics, image processing, computational geometry, geographic information systems, robotics [78Samet84], Computer-Aided-Design (CAD), computer vision, cartography and surgical planning [76Yau84]

Images, resulting from the reconstruction of projections in various directions, constitute a single CAT scan slice. Commercial CAT scanners produce as many as *277 slices (spaced at 1mm)* in one presentation of a region of human body. The voxel level information of sequence of planes thus stored may not be sufficient to generate a 3D model (to a resolution of $2^n \times 2^n \times 2^n$; n>6) for diagnosis by a physician. The situation then calls for interpolation to in between planes. The planes are represented by Z and the total number of planes are Z_{max} (sum of scanned and interpolated planes) for processing.

1.2.1 Formal Definition of Octrees

Octree represents the recursive decomposition scheme of representing 3D information, as a rooted tree in which every node has 8 or no children, stated formally.

Definition:-(1)

```
node  =record
       density       :real;
       child         :array[0..7]of reference to nodes;
       end;
octree =reference to node;
```

The recursive decomposition as outlined above results in a rooted tree with the following properties.

1. The tree is complete, in that every node has eight or no siblings.
2. A leaf node at level $k = 0$, (voxel level), represents a cubical sub-space of volume $2^k x 2^k x 2^k$.
3. The geometric position of the cubes corresponding to every leaf node is implicit in the tree representation.
4. The total space requirement to store the tree is of the order (7*L+1)/8 where L represents the total number of leaf nodes.

5. The siblings are ordered in the increasing order of X,Y,Z co-ordinates, during the post order traversal of the tree.

1.2.2 Representation of Octrees

1.2.2.1 Pointer based representation

The formal definition as given in definition (1) above, suggests one way of representing octrees, using pointers. Nine fields or more are required to represent a node, one field is to hold the information and eight fields are pointing to its children. The memory required to represent an octree using pointer type representation with 'L' external nodes is (8*(L-1)/7) nodes [79UnniA87]. The percentage of memory saving, by using an octree is dependent on the shape of the object and the orientation of the object in the universe cube. Consider a binary image, in which nodes are 'white' of the corresponding cubical space excludes the solid matter and 'gray' otherwise. Obviously the gray nodes are internal. A Pascal based definition is shown below:

```
pointer     =^node
node        =record
            child  :array[0..7]of pointer;
            type   :(black, white, gray)
            parent :pointer;
            end;
```

1.2.2.2 Linear Octrees

Gargantini [52Gargantini82] proposed a different method to represent an octree to improve the storage efficiency. In the case of a linear octree, only the leaf nodes (and only the black nodes in the case of binary image) forming the objects are stored. Thus only one field is required to identify a node of the octree. A node is represented in an encoded form, which absorbs the level of that node, size of the cube and position of the node of the cubical space.

Then a cubical space is given by

$Q = q_{n-1}8^{n-1} + q_{n-2}8^{n-2} + \ldots\ldots\ldots\ldots q_0\, 8^0$

where q_{n-1}, q_{n-2}, ...q_0 in sequence tells how to reach the nodes represented by the Q, from the root, and $q_i \in \{0,1,\ldots\ldots 7\}$.

If (x,y,z) are the co-ordinates of the cube, corresponding to the octal code <d_n, d_{n-2},....d_0, then

$z = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots\ldots\ldots + a_0$

$x = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots\ldots\ldots + b_0$

$y = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + \ldots\ldots\ldots + c_0$

where $d_i = 4a_i + 2b_i + c_i$ where a_i, b_i, c_i are either 0 or 1.

The eight pixels belonging to the same octant are grouped together, called *condensation*. The digit relative to the common octant are replaced normally with a marker 8 (remember octal code does not have 8), so that the sorted order of octal codes, after the post order traversal will not be distorted. The mixed octal codes generated using this method is simply termed as o-codes, and the representation scheme is called mixed-octal representation. Linear octree [52Gargantini82] representation is used for storing the image, in this thesis.

1.2.2.3 Hierarchical ordering in linear octree

The coding scheme for cubes as proposed by Gargantini [52Gargantini82] assigns a digit $d_i \in \{0,1,2...7\}$ such hat d is the link number $\{0,1,2...7\}$ of the octree at the level i. $0 \leq i \leq n$. However, if the homogeneity does not extent the coding upto level 0, but only up to a level k, $0 \leq k \leq n$, only the most significant n-k digits $\in \{0,1,2...7\}$ indicating link numbers. The remaining k digits are padded by 8 (or X, as proposed by Gargantini).

The linear hierarchical coding totally avoids the 8 padded to the code. The resultant codes are sorted level-wise so that

i) All codes at level l, $0 \leq l \leq$ n have (n-1) digits $\in \{0,1,2...7\}$ and

ii) All codes at a level l are sorted in increasing order.

1.2.2.4 Triangulation

In many applications, an even more fundamental problem than building the octree is acquiring the initial boundary data to form the boundary of the object being represented. One approach is to use a three-dimensional pointing device to create a collection of samples from the surface of the object. After the point data are collected, it is necessary to interpolate a reasonable surface to join them.

Interpolation can be achieved by triangulation. A surface triangulation in three-dimensional space is a connected set of disjoint triangles that forms a surface with vertices that are points in the original data set. There are many triangulation methods currently in use [161Bonneau98], in both two dimensional [80Watson84] [81Saalfeld87] and three-dimensional spaces [82Faugeras84]. They differ in how they determine which points are to be joined. For example, often it is desired to form compact triangles instead of long, narrow ones. However, the problems of minimizing total edge length or maximizing the minimum angle pose difficult combinational problems. Posdamer [83Posdamer82] suggests use of the ordering imposed by an octree on a set of points (eg. by bit inter-leaving [70Samet89, section 2.7 for more details]) as the basis for determining which points should be connected to form the triangles. Posdamer's algorithm uses an octree for which the leaf criterion is that no leaf can contain more than three points.

The K-structure [70Samet89, text reading] is a hierarchical representation based on triangulation rather than a regular decomposition. It is not built directly from the list of line segments in a polygonal map but rather than from a triangulation of the map. The notion of hierarchy in the k-structure is radically different from that of a quadtree in that instead of replacing a group of triangles by a single triangle at the next higher level, a group of triangles is replaced by a smaller group of triangles. There is no proper subset relationship between triangles at successive levels. In particular, a triangle at level k can be spanned by more than one triangle at level k+1.

1.3 Visualization of 3D information

Most of the 3D rendering techniques fall in three broad categories; Surface based rendering, Binary Voxel rendering and Volume rendering.

1.3.1 Surface Rendering

Surface based techniques, applies a surface detector to the sample array, then fits geometric primitives to the detected surfaces, and finally renders the resulting geometric representation. The choice of primitives and the scale at which they are defined are different in each technique. A common technique applies edge tracking on each data slice to yield a set of contours defining features of interest. A mesh polygon can then be constructed connecting the contours on adjacent slices.

The desirable characteristics of these techniques are inexpensive transmission, storage and compact geometric primitives. They also exhibit a high degree of spatial coherence, making rendering efficient [10Levoy90].

For clinical applications it is vital that the generated images represent reality. Ulf Tiede et al [6Tiede90] have investigated the quality of surface shading algorithms using computer simulated test objects. For the group of surface shading algorithms, they tested z-buffer gradient, and Marching Cubes [119Lorensen87] with two extensions. They discussed transparent visualization using transparent gray-level gradient shading as an example and found that gray-level gradient shading and Marching Cubes did not differ greatly, except for thin objects, where adaptive gray-level gradient shading was better. In contrast to the algorithm originally proposed by Levoy [137Levoy88], Tiede et al [6Tiede90] algorithm traverses the volume in a front-to-back manner and allows use of a stopping mechanism and the computing time was greatly reduced.

1.3.2 Binary Voxel Rendering

These techniques begin by thresholding the volume data to produce a three dimensional binary array. Voxels can be printed directly onto the screen in back-to-front [84Frieder85] (or front-to-back) order; rays can be traced from observer position through the data, stopping when an opaque object is encountered. The back-to-front (BTF) method displays entire solid objects composed of voxels without extracting the object surface first and with a minimum of pre-processing.

Since voxels have no defined extent, re-sampling becomes an important issue. To avoid a "sugar cube" appearance, some sort of interpolation is necessary. These techniques also require binary classification of the sample data, and thus suffer from many of the artifacts that plague surface based techniques.

Udupa and Odhner [85Udupa91b] propose a new data structure, which allows to interactively visualize, manipulate, and measure large 3D objects on general purpose workstations. This method requires seven to eight times less storage space and achieves three to five times faster computation compared to rectangular array representation of the binary scene.

1.3.3 Volume Rendering

Researchers at Pixar of San Rafel, California first demonstrated Volume rendering in 1985. Their techniques consist of estimating occupancy fractions for each set of material (air, muscle, fat, bone) that might be present in a voxel, computing from these fractions a color and partial opacity for each voxel, geometrically transforming each slice of voxels from object space to image space, projecting it onto the image plane, and blending together with the projection formed by previous slices [88Fuches, Levoy, Pizer 89].

Volume rendering is a technique for visualizing sampled functions of three spatial dimensions by computing 2D projections of colored semi-transparent volume [10Lovoy90]. Levoy addressed the problem of extending volume rendering to handle polygonaly defined objects. He proposed a hybrid ray-tracing algorithm. The rays are simultaneously cast through a set of polygons and a volume data array. The implementation uses several techniques that take advantage of spatial coherence to reduce rendering time. The first technique consists of constructing a hierarchical spatial occupancy enumeration (for instance, a complete pyramid of binary volumes, also called a complete octree) to speed up the search for non empty (non-transparent) voxels along viewing rays. The second technique uses an opacity threshold to terminate ray tracing adaptively. The third technique consists of casting a sparse grid of rays, less than one per pixel, and adaptively increasing the number of rays in areas of high image complexity. The algorithm computes colors and opacity directly from the scalar value of each voxel and renders the resulting volume by tracing viewing rays from an observer position through the data set.

Combining all three optimizations, savings of more than two orders of magnitude over brute-force ray tracing have been achieved for many data sets [86Levoy90b] [87Levoy90c].

The above technique is a variant of the Binary Voxel technique in which a color and a partial opacity is assigned to each voxel [88Fuches, Lovey, Pizer 89]. Images are formed from the resulting colored, semi transparent volume by blending together voxels projecting to the same pixel on the picture plane.

Researchers have developed many different methods to visualize the 3D geometry of anatomical structures. As an example, they have derived heuristic methods based on polygonal surface reconstruction from planar contours [89Keppel75] [90Fuches77]. Herman and liu [91Herman and liu79] have proposed cuberille method to display 3D objects from CT scans.

Volumetric rendering [92Fishman87] [93Scott87] have resulted in the generation of high quality images true to the initial data set. Anatomical areas where physicians have applied 3D imaging include nearly every part of the body, ranging from the skull to the ankle [94Kuhlman89] [95Kuhlman88] [96Magid90] [97Levin89] [98Vannier84]. In 30 to 50 percent of cases, 3D imaging has changed the choice of therapy significantly [99Fishman89].

Layered Depth Image (LDI), proposed by Shade et al [100Shade98], resembles volumetric representations. The main difference between LDI-based representation and 3D volume data are discussed in Dani et al [101Dani]. Curless and Levoy [102Curless96] and Wallin [103Wallin91] presented a volumetric method to extract an isosurface from range images. The goal of their work, however, was to build high detail models made of triangles/ polygons. There has also been work related to octree generation from range images [104Chien88] [105Connolly84] [106Li94]. However Octree that is generated in those methods is used to encode the space occupancy information. Each octree cell represents either completely occupied or completely empty voxel of the scene.

Layered manufacturing technologies have revolutionized the prototyping of complex geometric designs, but still employ traditional CAD tools. A voxel-based approach is under development in a modeling tool called G-WoRP [107Channdru95]. More details of G-WoRP can be refereed to Chandru and Manohar [108Chandru94]. They used the term layered manufacturing (LM) to denote any of the technologies that support the fast developing field of rapid prototyping [109Jacobs92] [110Burns93].

Two public-domain Volume Rendering packages are Bob and VolVis [107Channdru95].

Volume rendering suffers from a number of problems. High on this list is the technique's computational expense. Since all voxels participate in the generation of each image, rendering time grows linearly with the size of the data set. Another draw back of volume rendering is its lack of versatility. Many clinical problems require that sampled data and analytically defined geometry appear in a single visualization. Example includes superimposition of radiation treatment beams over patient anatomy for oncologist and display of medical prostheses for the orthopedist. Henry Fuches, Levoy and Pizar [88Fuches89] have developed two techniques for rendering mixtures of volume data and polygonally defined objects as discussed earlier.

Bining Guo [111Guo95] triangulate interval sets as α-shapes [112Edelsb83] [113Edelsb94] to achieve efficient rendering, as semi-transparent clouds, through tetrahedron projection [114Max90] [115Shirley90] [116Stein94]. Because interval sets are extracted in the object space, their visual display can respond to changes of the viewpoint or transfer function quite fast. The result is a volume rendering technique that provides faster, more effective user interaction with practically no loss of information from the original data. The hierarchical nature of multiscale interval sets also makes it easier to understand the usual complicated structures in scalar volumes.

Slusallek and seidal [117Slusallek95] proposed rendering architecture 'Vision' derived from a model of the physical rendering process, which is subsequently mapped onto an object-oriented hierarchy of classes. They gave a detailed description of the global lighting subsystem and show how algorithms for path tracing, bidirectional estimators, irradiance caching, hierarchical radiosity, wavelet radiosity, and wavelet radiance have been implemented within 'Vision'.

Surface rendering and volume rendering are the common methods in visualization [85Udupa91b] [111Guo95]. In surface rendering, one explicitly expresses the surface as a set of elements (points, line segments, polygons, triangles, patches, or cubes) then renders the surface using more or less standard computer graphics techniques. In volume rendering, one initially assigns opacity and a color to every data point (voxel) in a preprocessing step known as classification and then renders the volume by combining the colors based on the opacities. Surface and volume rendering techniques have some common properties as well as some disparate characteristics [118Udupa91a].

The Marching Cubes (MC) algorithm is a commonly used [6Tiede90] method for generating isosurface [119Lorenson87] [120Nig93]. The MC algorithm also generates an extensively large number of triangles (or polygon meshes in the object space) to represent an isosurface. Generally many triangles increase the rendering time, which is directly proportional to the number of triangles. Several attempts were made to reduce the number of triangles. The most notable, the decimation algorithm by Schroeder et al [121Schroeder92] substitutes the triangular mesh obtained by the MC algorithm with a simpler mesh generated from a subset of original vertices. Shu et al [122Shu95] reports an adaptive MC algorithm. The MC algorithm is first applied to cells of a given size, which is a power of 2 ($2^n x 2^n x 2^n$). If the approximating surface is not flat enough based on a curvature criterion, the initial cell is subdivided. The process continues until either the approximating surface is satisfactory or the initial cell is divided into 1x1x1 cells. Mountani et al [123Mountani94] have proposed descretized MC algorithm where the edge intersections are approximated by edge mid points. Wilhelms and Van Gelder [124Gelder92] report use of an octree data structure to enhance the MC algorithm. Raj et al [125Raj96] presents a decimation method to reduce the number of triangles generated by the MC algorithm.

Texture mapping is an easy way to achieve a high degree of realism in computer-generated imagery with little effort. Over the past decade, texture mapping technique have advanced to the point where it is possible to generate real time perspective simulations of real-world areas by texture mapping every object surface with texture from photographic images of these real-world areas. Weinhaus and Devarajan [126Weinhaus97] present a background survey of traditional texture mapping, known as Image Perspective Transformation. It then continues with a description of the texture-mapping variations that achieve these perspective transformations of photographic images of real-world scenes.

The main techniques for direct volume rendering are ray casting, data projection and Fourier volume rendering. Ray casting (eg. see [87Levoy90] generates high quality images but takes some minutes. Data projection methods include splatting [160Muraki94], cell projection and resampling compositing. The hierarchical splatting method by Laur and Hanrahan [127Laur91] and Westover [128Westover90] achieves interactive rate at the cost of image quality. Cell projection methods include, based on either cubes [129Wilhelms91] or tetrahedra [114Max90] [115Shirley90] are efficient when projected cells cover a significant screen area; otherwise the resampling synthesizing methods such as the 'Pixar' algorithm [36Drebin88] are faster. The "resampling compositing" method incorporating shear-warp factorization of the viewing transformation has near-interactive performance [130Lacroute, Levoy94]. Finally, for restricted volume illumination models, Fourier Volume Rendering has the potential to speed up the worst-case asymptotic complexity [131Totsuka93] [132Malzb93].

Ray tracing is a technique for rendering pictures from a 3D model by following the paths of simulated light rays through the scene. It supports various lighting phenomena, such as shadows, reflections, and refraction. One of the serious problems of ray tracing is that it requires a relatively large amount of computation time [133Kwon98], as it must perform the ray-object intersection test for each ray and each object in the scene. Techniques using the bounding volume and space subdivision have been two major approaches for improving the efficiency of ray tracing [134Arvo89] [135Foley90]. It is well understood that ray tracing is accelerated through two main techniques [136Reinhard98]: accelerating or eliminating ray/voxel intersection tests and parallelization. Acceleration is usually accomplished by a combination of spatial subdivision and early ray termination [137Levoy88] [138Kaufman91] [139Sobie94]. A new ray classification scheme that considerably reduces memory consumption while preserving its inherent time efficiency has presented in Kwon et al [133Kwon98]. As Whittened [140Whittened80] observed, the most time consuming operation is ray shooting, which is finding the first object hit by a given ray. Kwon et al [133Kwon98] has proposed acceleration of these operations.

Meinzer et al [141Meinzer91] developed Heidelberg Ray Tracing model for visualization of medical objects. Their model visualizes series of parallel plane Tomography images in 3D. It supports diagnosis and therapy by reducing 3D data sets to more easily understandable 2D images.

Ray tracing for volume visualization naturally lends itself towards parallel implementations [142Ma93] [143Muuss87]. The computation for each pixel is independent of all other pixels and the data structures used for casting rays are usually read-only. These properties have resulted in many parallel implementations. A variety of techniques have been used to make such systems parallel, and many successful systems have been built (eg. [144Vezina92] [145Schroder92] [146Muuss95]). Whitman surveyed these techniques [147Whitman95].

Parker et al [148Parker99] worked on brute-force ray tracing system for interactive volume visualization, runs on a conventional (distributed) shared memory multiprocessor machine. They have used several optimization including volume bricking scheme, and a shalow data hierarchy which uses three separate visualization algorithms: isosurfacing of rectilinear data, isosurfacing of unstructured data, and maximum intensity projection on rectilinear data. The system runs interactively.

Shum et al [149Shum97] presents a new approach to free-form object modeling from multiple range images. In contrast to the registration of successive views sequentially, they propose an integral approach, which reconstructs statistically optimal object models by simultaneously aggregating all data from multiple views into a weighted least-squares formulation.

D.R. Ney and E.K.Fishman [150Ney91] designed a drawing program for 3D data sets; MPR (Multiplanner Reconstruction Editor) edit lets one interactively create shapes that define volumes of interest in images of medical data.

High quality image compression algorithms are capable of achieving transmission or storage rates of 0.3 to 0.5 b/pixel with low degradation in image quality. In order to obtain even lower bit rates, Rayn et al [151Ryan96] relax the usual rms error definition of image quality and allow certain "less critical" portions of the image to be transmitted as texture models. These regions are then reconstructed at the receiver with statistical fidelity in the mid-to high-range spatial frequencies and absolute fidelity in the low pass frequency range. This hybrid spectral texture modeling technique takes place in the discrete wavelet transform domain. In this way Rayn et al obtained natural spectral texture models and avoided the boundary blending problems usually associated with polygonal modeling. Rayn et al [151Ryan96] describes the complete hybrid compression system with emphasis on the texture modeling issues.

Vector Quantization [152Nasrabadi88] [153Cohn94] [154Li95] [155Kim97] and Wavelets [156Eric95] [157George97] [66Raghu98] are data compression techniques to reduce the information losses and to enhance the data retrieval. A very high data compression ratio is obtained.

In some earlier work, Muraki [158Muraki93] applied wavelet transforms to volume data to achieve compact approximations. For ray casting volume data, Westermann [159Westernann94] demonstrated how to work with wavelet coefficients directly so that large volumes can be rendered with reasonable penalty in time. Muraki

[160Muraki94] also proposed a difference of Gaussians (DOG) wavelet for exposing the multiscale edges in volume. On the positive side, DOG wavelet is spherically symmetric and thus beneficial to some data projection methods (eg. Splating). On the negative side, approximating volume data with DOG wavelet requires an expensive iterative process; to achieve the same goal with derivative wavelets, a single wavelet projection suffices.

Wavelet-based methods have proven their efficiency for the visualization at different levels of detail, progressive transmission, and compression of large data sets. The required core of all wavelet-based methods is a hierarchy of meshes that satisfies subdivision-connectivity: This hierarchy has to be the result of a subdivision process starting from a base mesh. Examples include quadtree uniform 2D meshes, octree uniform 3D meshes, or 4-to-1 split triangular meshes. In particular, the necessity of subdivision connectivity prevents the application of wavelet-based methods on irregular triangular meshes. Bonneau [161Bonneau98] introduced "wavelet-like" decomposition that works on piece-wise constant data sets over irregular triangular surface meshes. The decomposition/reconstruction algorithms are based on an extension of wavelet theory allowing hierarchical meshes without subdivision-connectivity property. Author, Bonneau, claims that this approach allows exact reconstruction of the data set, even for non regular triangulations, and it extends previous results on Haar-wavelet over 4-to-1 split triangulations.

The voluminous nature of 3D data prompted research in new techniques for economizing both storage space and processing time. Fractal techniques based on iterated function systems [162Hutchinson81] [163Barnsley85] have been successfully applied to the compression of one-dimensional signals [164Barnsley89] [165Vines93] and two-dimensional images [166Jacquin92] [167Fisher92] by finding a fractal representation that models the original signal as closely as possible, and storing (and further compressing) the model instead of the original data.

In contrast to the typical notion of fractal methods, block based fractal compression technique perform very well on smooth edges, and rather poorly on textured regions. Whereas fractal image compression of 2D medical data, such as X-rays, may blur the textured regions of most interest, fractal volume compression of 3D medical data, such as CT scans, represents edges sharply and yields well-defined surfaces.

Cochran et al [168Cochran96] proposes fractal volume compression as a competitive alternative to other volume compression methods. The extension of fractal image compression to volumetric data is trivial in theory. However, the simple addition of a dimension to existing fractal image compression algorithms results in infeasible compression times and noncompetitive volume compression results. Cochran et al extends several fractal image compression enhancements to perform properly and efficiently on volumetric data, and introduces a new 3D edge classification scheme based on principal component analysis. Numerous experiments over the many parameters of fractal volume compression suggest aggressive settings of its system parameters. At this peak efficiency, fractal volume compression surpasses vector quantization and approaches within 1dB PSNR of the discrete cosine transform. When compared to the DCT, fractal volume compression represents surfaces in volumes exceptionally well at high compression rates, and the artifacts of its compression appear as noise instead of deceptive smoothing or distracting ringing.

Kim and Park [169Kim96] propose a coding algorithm for still images using Vector Quantization (VQ) and fractal approximation, in which Vector Quantization approximates low-frequency components of an input image, and its residual is coded by fractal mapping.

1.4 The outline of the approach.

The central idea of the thesis is to develop algorithm for representing octrees generated out of CAT scan slices. It is felt that for a medical imaging system the synthesis of octrees from CAT scan slices would be the right starting point. We proposed a space and time efficient algorithm that looks at CAT scan slices plane by plane and progressively build the octree [75Sojan97]. We have used the linear octree representation [52Gargantini82]. We also proposed a parallel implementation of octree generation algorithm by ordering the linear octree code on the basis of volume [41Unni88]. Needless to say, any parallel algorithm should assume a basic interconnection of processing elements and this has motivated us to process an algorithm based on Transputer for high-speed implementation of octree generation [60Sojan98]. It is shown that the algorithm can be effectively implemented on a network of Transputers.

Noting that the representation of average density within a cubical space leads to a "blocky" (i.e data contained in the cubical sub-space of the octree can not be always equal to a constant value.) nature in the re-constructed solid, we addressed the issue of representing octrees with reduced information loss. It is demonstrated that the wavelet transformation can be effectively combined with the algorithm for synthesizing the octrees from CAT scan slices [170Sojan01]. Though we have explored the possibility of using vector quantization (VQ), the relative difficulty in handling vector quantization technique along with the conversion dissuaded us from incorporating the same during the octree generation.

Since the processing algorithm using octree are plenty in literature, we did not venture into proposing any new approach in this area. However, while attempting to display the solids represented by solids, existing algorithm based on Marching cubes [119Lorenson87] [125Raj96] [123Montani94] produces the solid profile without showing the hidden details. Following an approach similar to Levoy [10Levoy90], we have shown that the denser regions inside the solid matter can be brought out using transparency shading. Such a rendering technique would help in medical imaging, where not the outer solid profile, but the inner details are relevant. Finally the thesis concludes with directions for future work.

1.5 Conclusion

The thesis is organized as follows: Chapter 1 gives an overview of the representations of 3D information followed by discussion on the octree representation.

Parallelism of bottom-up raster to octree conversion algorithm were realized, under concurrency on VAX-VMS and also on Transputer. First half of Chapter 2 brings out the implementation details of concurrency model, with the creation of sub-process by the scheduler and synchronization mechanism using different flags for concurrent execution. Second half narrates the parallel implementation model using Transputer; "Farmer" node farms all tasks to "worker" nodes and the final result is harvested by the "Farmer" itself. Communication between processors was achieved by "shared memory" and "message passing" techniques. Each worker node is assigned a specific region in memory.

An elegant approach is suggested in Chapter 3 for minimum information loss, using wavelets. Display strategies of octree-encoded images are discussed in Chapter 4. The conclusion of the thesis along with directions for future work is brought out in Chapter 5.

Chapter 2

Sequential and Parallel implementation of Octree Generation Algorithm

Abstract

In an attempt to derive efficient algorithm for building octrees from CAT scan slices, it is shown that the octree can be progressively built by considering the CAT scan slices plane by plane. After an initial demonstration of algorithm under concurrency detailed approach for building octree on Transputer network is addressed. The time and space complexities of the algorithm are also derived.

In view of the fact that the octree encoding results in large amount of data compression, while dealing with 3D information, the present Chapter addresses the basic properties of the octree data structure and the parallel implementation using concurrent features in VAX-VMS. We also illustrated how the parallelism are realized using Transputers at the end of the Chapter.

The idea of hierarchical organization of information in octrees suggests itself the use of pointer structure in which $2^n x 2^n x 2^n$ every node has eight (or nil) pointers to siblings and one node to the parent. The organization has been shown to take space proportional to 8*L-1/7, where L is the total number of external nodes in the tree [79UnniA87]. An alternate representation called linear octrees stores the octree as a sorted array of numeric codes, where each code is an octternary number of n digits D (0, 1,2...7) *n* being the resolution factor. The octternary code encodes the leaf nodes of the tree in respect of the position of leaf nodes in the tree and the geometric position of the corresponding cube in the image space. The present report has made use of linear octree [52Gragantini82] representation in the implementation attempted.

Representation of octree and few rendering techniques were seen in the last Chapter.

Consider two lists L1 and L2 of codes, as in Fig.2.1, corresponding to two adjacent planes (say (0,1), (2,3)...etc.). The lists L1 and L2 are necessarily in sorted order.

Let

$L1 = \{0_1, 0_2, ..., 0_{i-1}, 0_i, 0_{i+1}, 0_{i+2}, ..., 0_n\}$

$L1 = \{P_1, P_2, ..., P_{i-1}, P_i, P_{i+1}, P_{i+2}, ..., P_m\}$

The codes 0_{i-1}, 0_i, 0_{i+1} and 0_{i+2} may be represented as under:

$0_{i-1} = <d_{n-1}\ d_{n-2} .... d_1 0>$

$0_i = <d_{n-1}\ d_{n-2} .... d_1 1>$

$0_{i+1} = <d_{n-1}\ d_{n-2} .... d_1 2>$

$0_{i+2} = <d_{n-1}\ d_{n-2} .... d_1 3>$

and similarly

$0_{i-1} = <d_{n-1}\ d_{n-2} .... d_1 4>$

$0_i = <d_{n-1}\ d_{n-2} .... d_1 5>$

$0_{i+1} = <d_{n-1}\ d_{n-2} .... d_1 6>$

$0_{i+2} = <d_{n-1}\ d_{n-2} .... d_1 7>$

where $d_i \in \{0,1,2........7\}$, $0 < i < n$

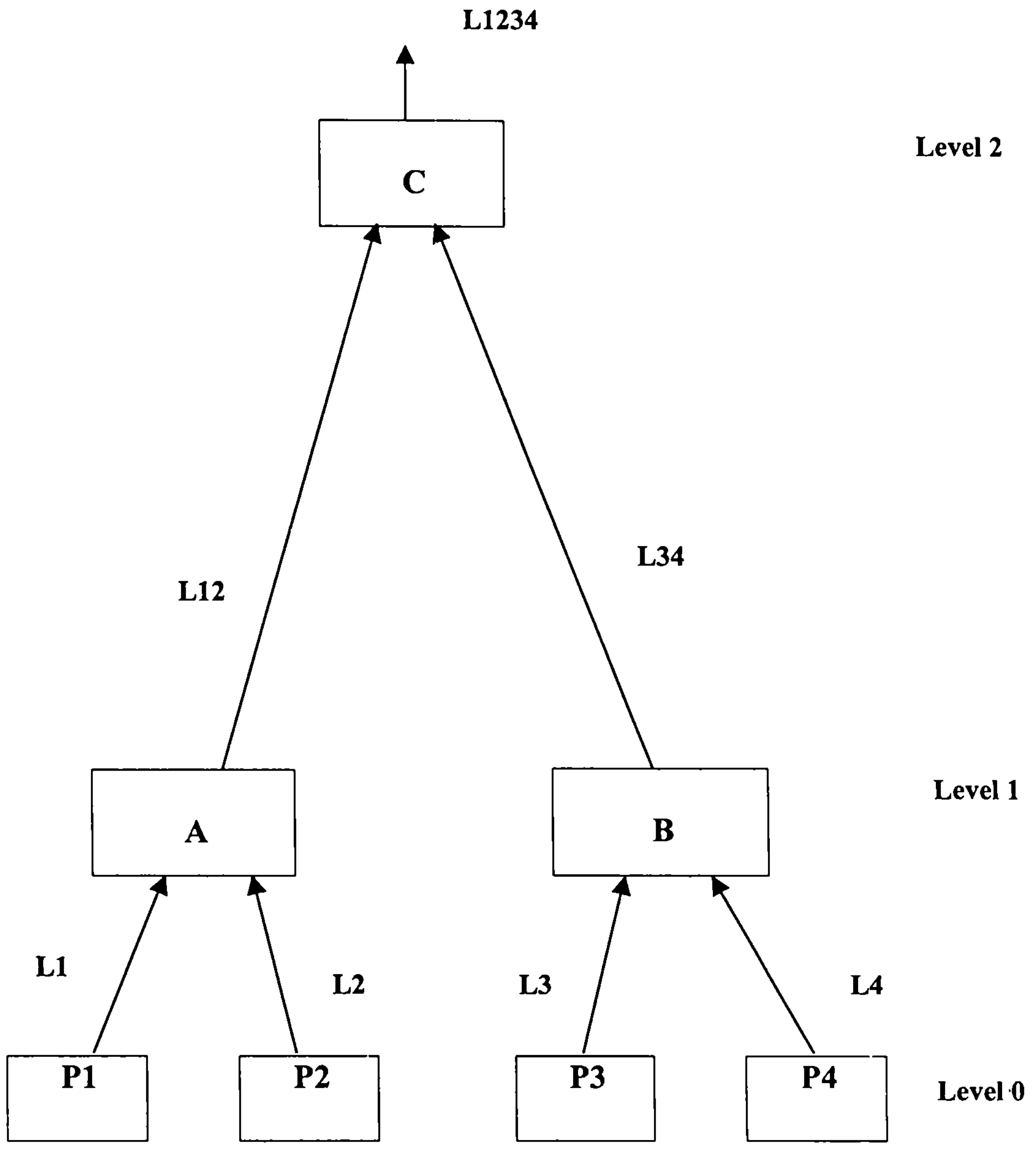


Fig.2.1 Illustration of the bottom-up recursive algorithm.

*List L1 processed by Process P1 and the list L2 processed by Process P2. The condensed codes of lists L1 and L2 are merged to L12. Similarly the lists L3 and L4 create list L34. The final result after condensation and merging is available in list L1234

These codes occupy the same octant at level 1, with each code representing the cube occupancy in a sub-octant at level 0. All the 8 codes can be represented and replaced by

$$O^l = <d_{n-1}\ d_{n-2} \ldots\ldots\ldots\ldots d_1 8>$$

The 8 correspond to the label X of Gargantini [52Gargantini82] and indicate that O^l represents a cubical volume of 2x2x2. The 8 appended ensure that the partial ordering is not disturbed. The obvious method then is to merge the two lists with possible condensation. However the size of the merged list could be less than the sum of the sizes of the two lists used for merging.

Reasoning further, two lists will be produced after pair-wise merging from 4 planes. Each of the lists will have codes of level 0, and or level 1 produced by condensation. 4 codes of level 1 from each of the two lists (numbering totally to 8) can condense to a code of level 2. Proceeding thus, one can think of a hierarchy of operations involving

(i) Generating codes from planes at level 0 and

(ii) Merging and condensing at level > 0

The resulting algorithm is presented with a recursive formulation in the section to follow:

2.1 Bottom-up recursive algorithm

The algorithm shall be explained by recursive formulation as in Fig.2.1 and Fig.2.2.

The procedure *BUILD_OCTREE* is invoked recursively from *n*th level with the following parameters.

```
PROCEDURE BUILD-OCTREE (LEVEL, L, HALF)
BEGIN
        IF LEVEL = 0 THEN
            GENERATE_CODE INTO L;
        ELSE
            BUILD_OCTREE(LEVEL-1,L1, LEFT_HALF);
            BUILD_OCTREE(LEVEL-1, L2, RIGHT_HALF);
            MERGE_CONDENSE((LI, L2) TO L);
END;
```

Fig.2.2 The recursive generation of octree form CAT slices.

BUILD_OCTREE (LEVEL, L, HALF) as in Fig 2.2, returns a sorted list of o-codes in list L. The procedure recursively explores the left and the right halves to lower levels until it reaches the level 0. At level 0, the sorted list of o-codes for level 0 corresponding to the plane encountered is generated. During the exploration of the two halves, the planes from 0 to $(2^n/2)-1$ and $(2^n/2)$ to 2^n-1 are taken care of by left and right invocations respectively. Eventually such recursive invocations would lead to the procedure encountering a plane numbered z, $0 \le z \le 2^n$

At levels > 0, the left and right invocations would fetch two lists L1 and L2. These lists, being individually sorted, are merged with possible *condensation*.

2.2 Practical aspects of implementation

It can be seen from the above discussion that each merging could result in the length of lists increasing to a size = $|L1| + |L2|$, where L1 and L2 are the merged lists. It can also be seen that if a code is produced at a level $l > 1$ due to condensation, it would have been produced only from 8 condensable codes at level l-1, and only the codes of cubes of size $2^{l-1}x2^{l-1}x2^{l-1}$ would have been taken into account. The consequence is that at every levels $0 < l < n$ only the lists of condensed code need to be sent to a higher level for possible condensation at level $l+1$. The lists L1 and L2 with the condensable codes removed can be merged at every level to form a separate list L to be returned from level l. The condensed list can be returned in another list L^1 to be considered for condensation at level $l+1$. To accomplish the arrangement as above, we have chosen to extent the idea of linear hierarchical quardernary coding to 3D.

2.3 Linear Hierarchical coding

The coding scheme for cubes as proposed by Gargantini [52Gargantini82] assigns a digit $d_i \in \{0,1,....7\}$ such that d_i is the link number (0,1...7) of the octree at the level $0< i< n$. However, if the homogeneity does not extend the coding upto level 0, but only upto a level k, $0< k < n$, then only the most significant $n-k$ digits $\in \{0,1...7\}$ alone indicate the position of the leaf node in the tree. The remaining k digits are padded by 8 (or X, as proposed by Gargantini), to indicate the volume of the cube represented.

The linear hierarchical coding totally avoids the 8 padded to the code. The resultant codes are level-wise sorted so that

1. all codes at level l, $0 < l < n$ have $(n-l)$ digits $\in$ {0,1..7) and

2. all codes at a level l are sorted in increasing order.

2.4 Handling condensation with linear hierarchical coding

Let two lists L1 and L2 at level 0 are taken up for merging and possible condensation.

Let

$$L1 = \{01, 02, \ldots\ldots\ldots 0_{i-1}, 0_i, 0_{i+1}, 0_{i+2}, \ldots\ldots\ldots 0_n\}$$

$$L1 = \{P_1, P_2, \ldots\ldots\ldots P_{i-1}, P_i, P_{i+1}, P_{i+2}, \ldots\ldots\ldots P_m\}$$

The codes $0_{i-1}, 0_i, 0_{i+1}, 0_{i+2}$ and $P_{i-1}, P_i, P_{i+1}, P_{i+2}$ may be assumed to condense to a single code 0^l at level *l*, obtained by deleting the least significant digit from any one of $\{0_{i-1}, 0_i, 0_{i+1}, 0_{i+2}, P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$. The codes similar to O^l, generated out of condensation of L1 and L2, are put into a list L(1). Obviously L(1) will be in sorted order. The lists L1 and L2 are relieved of $\{0_{i-1}, 0_i, 0_{i+1}, 0_{i+2}$ and $P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$ and then simply merged to a list L(0). At level *l*, (node (A) in Fig.2.1), lists L(1) from left half and a similar list from right half alone are taken up for condensation, while the two L(0) lists from each half are simply merged. Along the lines explained above, the two lists L(1) from each half, results in a list L(2) after condensation. The uncondensed codes of the two L(1) lists are merged to a single list L(1). The merging and condensation proceeds thus up to the level n. At each level *l*, only a list of level *l* and of a relatively smaller size is examined for condensation. Straight merging take place between lists of level $\leq l$.

2.5 The implementation

The data structure of the implemented model is stated formally below in Def.2:

To begin with the codes for all planes are contained in level zero. This data is condensed and merged as in Fig.2.4 As the condensation progresses; the condensed o-codes of the adjacent pair of planes are posted to the higher level, thus reducing the requirement of the memory to half. The numbers of condensed o-codes are not only proportional to the number of o-codes at the lower level, but also to the homogeneity inside the solid profile. It would therefore be desirable to have a dynamic allocation of space, as against the static representation given above. The formal algorithm is illustrated in Fig.2.3.

Def.2:

```
node = record
        ocode     :integer;
        intensity :integer;
        end;
plane_list    :array [0..$X_{max}$*$Y_{max}$] of node;
level_info    :array [0..$Z_{max}$] of plane_list;
solid         :array [0..level_max] of level_info;
```

The bottom-up iterative version of implemented algorithm is illustrated in Fig.2.3 and the structure is illustrated in Fig.2.4.

The execution of the program starts by generating the frame (optional) [Fig.2.3] for the solid. The corresponding o-codes are then generated and sorted plane-wise along with its intensity. This is required, to locate quadrants in each plane and then to find out the octant, to which it belongs. Condensation is progressively carried out for planes at different levels hierarchically. Typically the o-codes at level l is condensed and posted to level $l+1$. The voids (free space) created by condensation at level k, are removed by merging the pair of planes at level l and keeping the merged result a single list. (The value of o-code at 0th location of each "plane_list" is reserved to indicate its length, and value of intensity at 0th location of each pair contains the length of the corresponding pair after condensation and merging.). These sets of

```
START
{
    DO PLANE = 0 to ZMAX,1
    {
        GENERATE_FRAME(PLANE);
            This procedure is invoked only when we generate the data within the
            program. The data is sent to a file. This data is available from the
            scanner, thus this stage is superfluous.
        GENERATE_OCODE(PLANE);
            The intensity/density information is read and the corresponding ocode
            is calculated. Now each voxel information has ocode and density.
        SORT-OCODE(PLANE);
            Row-wise sorted ocodes are then merged to form a single list, which
            represents a plane, helps to locate quadrants
    }
    END DO
    LEVEL_MAX = LOG (ZMAX)/LOG (2)
    DO LEVEL =0, LEVEL_MAX
    {
        DO PLANE = 0, ZMAX/2**LEVEL,2
            CONDENSE_OCODE(LEVEL, PLANE);
          Pairs of sorted planes at level k are scanned for possible condensation, and
          the condensed result is posted to higher level (ie, k+1), and merge the pairs
          of planes at level k to form a single list.
        END DO
    }
    END DO
 DO LEVEL = 0, LEVEL_MAX
    COPY_TO_LINEAR_ARRAY(LEVEL)
        copying the pairs of planes to linear array for an easy merging
 END DO
 DO LEVEL = 0,LEVEL_MAX
    LEVEL_WISE_MERGING(LEVEL);
    The ocodes are merged within each level.
 END DO
 FINAL_MERGE;
    Final merging is carried out between levels. The result is a mixed-octal-
    representation of the solid matter
 WRITE_OCODES; The result is copied to a secondary memory
}
STOP
```

Fig.2.3 Iterative version of bottom-up algorithm.

single list is merged plane-wise and then level-wise. The single list formed by condensation and merging is a mixed o-code representation of the solid, along with its density absorption values (or intensity).

The Fig.2.4, illustrates the structure and the process of progressive condensation and merging. As was indicated earlier, the 0th element of each array carries two counts,

1) The size of the array generated or produced from condensation (.node(0).code)
2) The size of the array after merging (.node(o).intensity). At any level, the arrays are kept contiguous pair-wise, after merging.

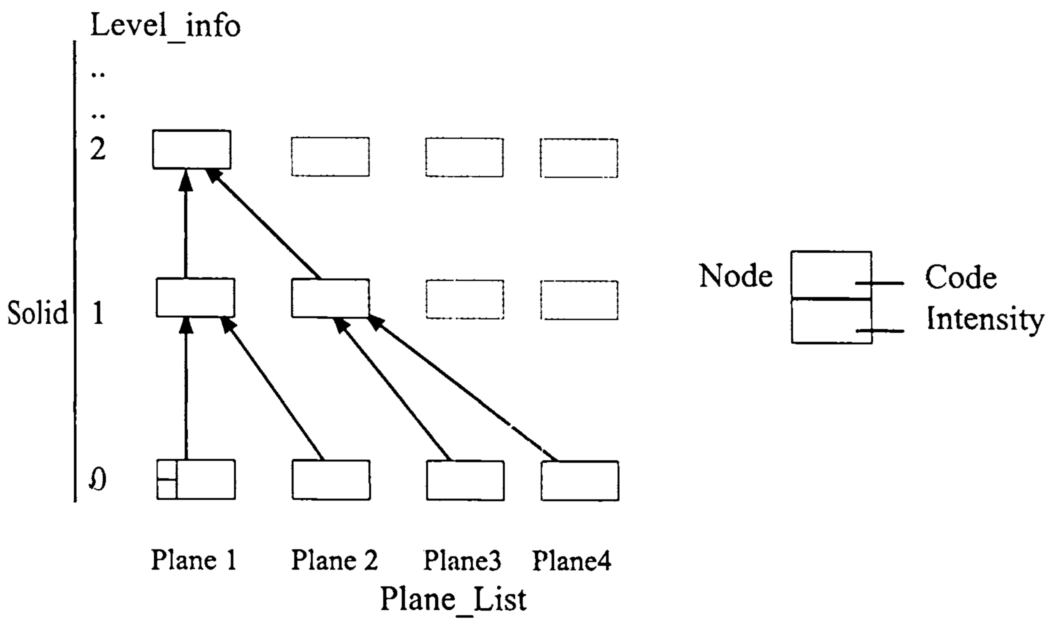


Fig.2.4 Illustration of the execution of the algorithm in Fig.2.3.

Data structure defined in Def.2

Condensation and merging of pair-wise planes (list) are demonstrated. Level 1 requires half the size of the memory than level 0. i.e. Two adjacent planes are condensed and merged to one plane in the next higher level.

The algorithm was tested on a simulated set of slices. The slices were generated with a circle, keeping two holes in the middle, such that the circles of varying radii resulted in a sphere. The intensity values were arbitrarily assigned to give one value to the spherical shell. Fig.4.3 displays the progressive stages of generation of the solid.

We demonstrated that the universal cube of size $2^n x 2^n x 2^n$ is subdivided into eight disjoint sub cubes of size $2^{n-1} x 2^{n-1} x 2^{n-1}$. Thus each sub cube can be independently processed, either concurrently or in parallel, with increased speed of execution. This will be discussed in the next Chapter.

2.6 Parallel Execution Of The Bottom -Up Algorithm

It would have appeared from the previous sections that the time taken to generate the octternary code even for a 32x32x32 Tomographic solids are quite high. Though one can keep off the time taken for the generation of the slices from the total time required to generate the codes, the requirements to access each voxel could lead to large amount of time during the generation of codes. On the other hand, the codes in disjoint, but contiguous planes at positions 2^p to 2^p+2^k, $0 \leq p \leq 2^n-1$, $0 \leq k \leq 2^n-1$, can be generated by independent processes, such that one process can finally handle the last few stages of condensation and merging.

2.7 The Computational Model

Fig.2.5 shows the computational model for N=32. There are 4 processors, each handling 8 planes. A given processor can then execute the algorithm described in Fig.2.3 generating codes of level 0, 1, 2 and 3. Following the hierarchical coding

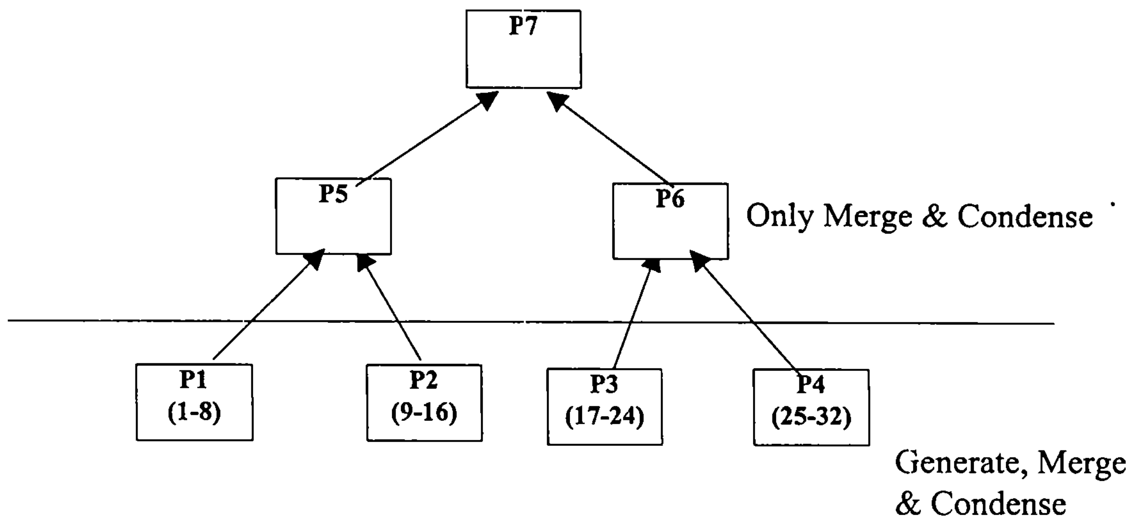


Fig.2.5 Parallel computing model for building octrees, bottom-up

***Each processor processes the o-codes for the specified region only and return the result to the parent for possible condensation, merging and final processing.**

mentioned earlier, each processor could produce a sorted list of codes corresponding to levels 0,1,2 and 3 generated from the 8 planes under consideration. Since the codes of level 3 have a potential to merge with the codes of level 3 from the neighbouring set of planes, processor P1 and P2 (of Fig.2.5) would return the respective sorted lists of level 3 codes to the processor P5. P5 would then merge/condense the level 3 codes, producing two lists viz. of level 3 and level 4. A similar operation takes place in the other half (corresponding to planes 17 to 32). A different processor P6 may be scheduled to generate the list of level 3 and 4. But in view of the fact that P5 is dealing with relatively lesser number of codes, the task of P6 can also be done by P5 with a little bit of sequencing.

2.7.1 Parallel execution of Bottom-up Algorithm on VAX-VMS

Our model [75Sojan97], has 4 independent processors to handle the generation of codes from planes, and these 4 processors independently merge and condense to levels 0, 1, 2 and 3. The processors send the codes of level 3 to a parent processor P5 (Fig.2.6) When two lists are received by P5 from adjacent processors (say P1 and P2 or P3 and P4), P5 schedules a processor P6 to handle the merging and condensation of codes of level 3, to progressively generate codes of levels 3, 4 and 5. We have also chosen to direct all the processors P1 to P4 and P6 to keep their processed lists (i.e., those lists from which no further condensation is possible) level wise sorted in their

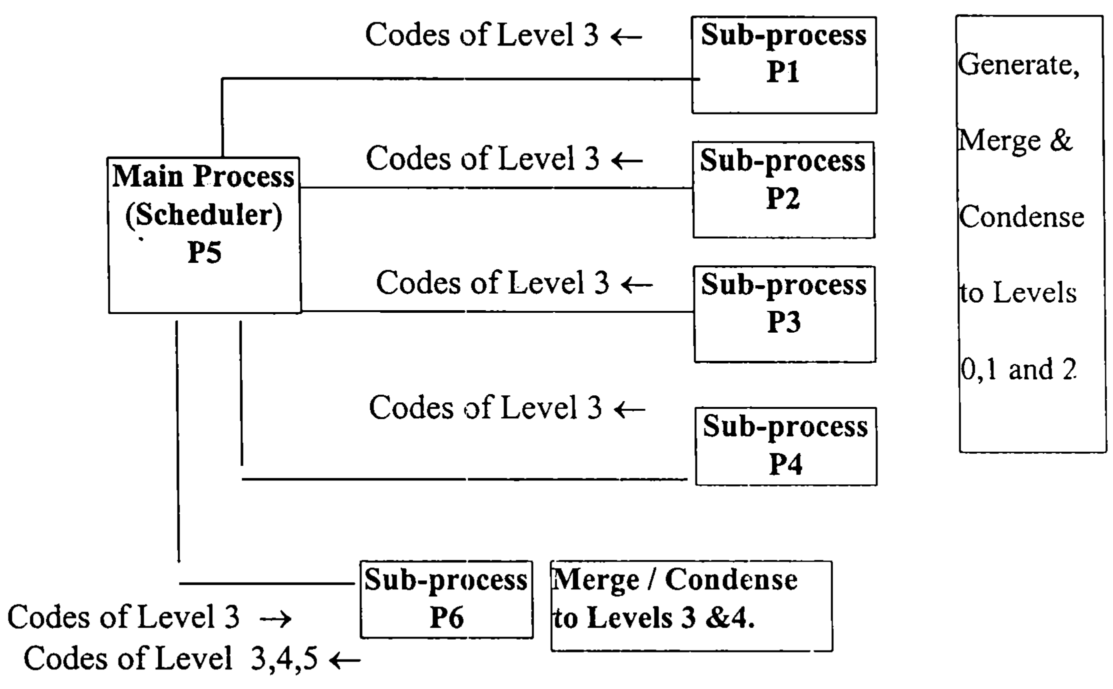


Fig.2.6 The implemented Model, on concurrent system (VAX/VMS).

The main Process P5 assigns the tasks to sub-Process P1-P4. Each Process handles 8 planes each and executes concurrently. Codes are Generated, Merged and condensed in the Sub-process upto Level 3. The Main Process P5 assigns all level 3 codes to P6 for merging/condensation of level 3 codes. P5 finally merges all the codes at all levels. Synchronization of the Processes is carried out as shown in Fig.2.7 using Flags.

respective private memories. The scheduler P5 can then finally merge all the lists to get the partial ordering, as desired in linear octrees [52Gargantini82]. We have demonstrated that the concurrent execution of the various processing modules of the octree generation algorithm leads to an improvement in the turn around time over the sequential program. This promises a marked improvement in the execution time, if parallel hardware is available to support the computation. The model chosen also establishes good processor utilization strategy where more processors are assigned to do the tedious jobs in parallel. Since the information to be handled at the higher levels is relatively less, the lesser number of processors, for higher level is a judicious choice.

The synchronization of all processes was implemented by passing different flags as shown in Fig.2.7. The sub process P1 sets the FLAG0 to signal the scheduler P5, so that P1 can continue the execution and P5 can schedule the next subsequent jobs P2 to P4 in sequence. FLAG0 thus guards the critical region of plane specification. When the lower level *condensation* upto the level 0,1,2 are completed, each sub processes sets the respective status bit & sets FLAG1 to signal the scheduler P5, and proceeded for lower level merging. Whenever the scheduler gets the signal FLAG1 from sub-processes P1 to P4, it checks the status bit for adjacency of regions .for higher level condensation by sub-process P6. FLAG2 signals P6 for starting condensation, FLAG3 signals back to the scheduler P5 on completion. This simulates a near strict parallelism, as the merging for a region of planes at different levels is carried out by different processes concurrently.

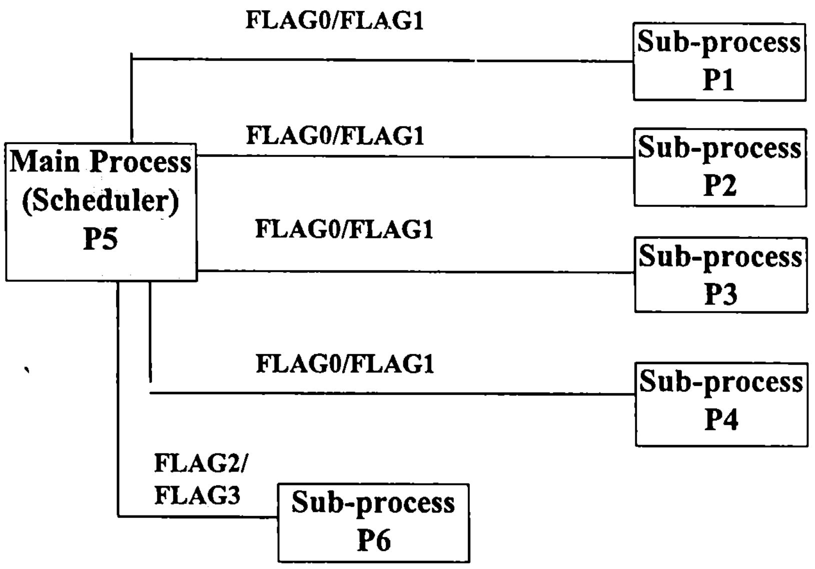


Fig. 2.7. Illustration of synchronizing the model

The main Process P5 assigns the tasks to sub-Process P1-P4, as shown in Fig.2.6. Synchronization of the Processes is carried out as shown in Fig.2.7 using Flags.

We have chosen to direct all processes to flush out the level wise sorted lists of codes to secondary memory. The waiting scheduler P5 (Fig.2.6) completes the work of merging all the level wise sorted lists taken from secondary memory to a single list of partial ordering.

We have implemented both the sequential and the parallel version of the octree encoding algorithm. The CAT slices used were however synthesized to have two cylindrical regions in the sphere. The 3D image of size 32x32x32 could be represented by cubes. Images of larger size could definitely be handled, with a judicious management of storage space, dynamically and if required in secondary memory. The parallel version of the algorithm was realized using the Concurrency Constructs available in VAX VMS operating system.

2.7.2 Parallel execution of Bottom up algorithm and porting to Transputer

The architecture of Transputer has been designed such that a single Transputer can either execute a single sequential process or several concurrent processes sharing processor time. Concurrent processes can also be distributed among interconnected Transputers for parallel execution on multiple Transputers. The process model of computation implemented on Transputers provides explicit input and output commands for receiving and sending data between concurrent processes. The medium through which communication between two processes takes place is called 'channel'. The channel is a unidirectional, point-to-point communication link between two processes and the channel ensures synchronized communication [171Sinha94].

Attempts have been made to extend the simple von Neuman design to multi-processors for parallel operation using a single bus for MIMD machine. However, a number of problems arise when using a single-bus multiprocessor. Adding more PEs to the data bus, after an initial improvement, causes a degradation in the overall performance due to the increased competition for use of the shared data bus (bus-connection). This results in extended 'idle' times for each extra processor as they wait to gain access to the bus. Since in the von Neumann model only one processor may have access to the bus at any time, this condition is known as the von Neuman bottleneck [172Dettmer85]. The point-to-point configuration supported by the Inmos Transputer, through four link interface units per PE, facilitates inter-processor communication and avoids the potential communication bottleneck of the single-bus system. The INMOS Transputer is a general purpose, single chip, high-speed microprocessor that is a member of the MIMD class of parallel architectures.

The two main classes of architecture, which dominate the parallel processing market, are the shared and distributed memory systems. The shared memory architectures are characterized by a shared address space between the processing elements [173Gottlieg83] [174Chang79]. Distributed memory architectures, by contrast, involve processors with their own local storage, which communicate with each other through message passing [175Pase88].

Mahamed Ada [176Ada97] presents the development and the performance of a novel bus-based message passing interconnection scheme which can be used to join a large number of INMOS Transputers via their serial communication links [177INMOS89] The main feature of this architecture is that it avoids the communication overhead which occurs in systems where processing nodes relay communications to their neighbours. It also produces a flexible and scaleable machine whose attractive characteristics are its simplicity and low latency for large configurations. He has shown that this architecture is free from deadlock, exhibits much smaller latency than most directly connected Transputer networks and has a scaleable bandwidth, in contrast to other bus topologies.

Transputers have found vast variety of applications in real time control [178Irwin92].

The geometric approach to simulation on a distributed memory parallel computer requires that the problem be partitionable into separate sub problem, where each sub problem can be solved essentially independently. The simulation method [179Ponton90] used involves splitting the distillation column into sections and placing each one on its own Transputer. The sections are then solved separately. Each section requires information from its nearest neighbours, this being provided by the inter-processor communication links.

The bottom-up approach of building Octree from CAT scan slices can be speeded up by allocating groups of planes (Say 8 planes for a 32x32x32 grid) to each "worker" nodes of a Transputer [60Sojan98] as shown in Fig.2.8 using balanced "tree" topology. A processor managing P planes ($P = 2^p$; p is an integer) can be farmed to different worker nodes [180Stephen94] [181Day92] P_1, P_2, P_3 and P_4 for generating the tree up to the level p or o-codes for level 0,1,..p independently and communicate with their parent (/Farmer) P_5, P_6 or P_7 to generate the codes for higher levels. Condensed codes of level 3 alone are passed to their respective parents P_5 and P_6 for further condensation and merging. Finally the farmer P_7 will do the final merging to produce a single list of sorted o-codes. The above scheme of farming the work can be implemented on a network of 4 Transputers connected as shown in Fig.2.9. The i-o access time, real time rendering of voxel level information, communication harness, synchronization and data retrieval time are minimized in this Transputer model [60Sojan98] as against the earlier reported sequential algorithm [75Sojan97], since it communicates with each other via their communication links and the memory shared between worker and farmer. T9000 virtual channel processor would allow any number of logical channels to be multiplexed into four physical communication links [182Barrett92].

Processor I in Fig.2.5 farms the jobs to processors II, III and IV, at the same time sharing the job of generation, condensation & merging at level 0, 1, 2 and 3 there by balancing the loads. Level 3 condensed codes from processor II and IV moves to processor I and III respectively, to condense and produce any possible level 4 codes. Sorted lists of codes of levels 0, 1, 2 and 3 move from processor II to I and IV to III, to be merged into separate list of codes of level 0, 1, 2 and 3. Final harvesting of codes of levels 0, 1, 2, 3 and 4 from processor III to I results in the codes of level 0, 1, 2, 3 and 4 merged level wise. The codes are then merged by the farmer to produce a single sorted list representing the object. The situation can be illustrated as shown in Fig.2.10.

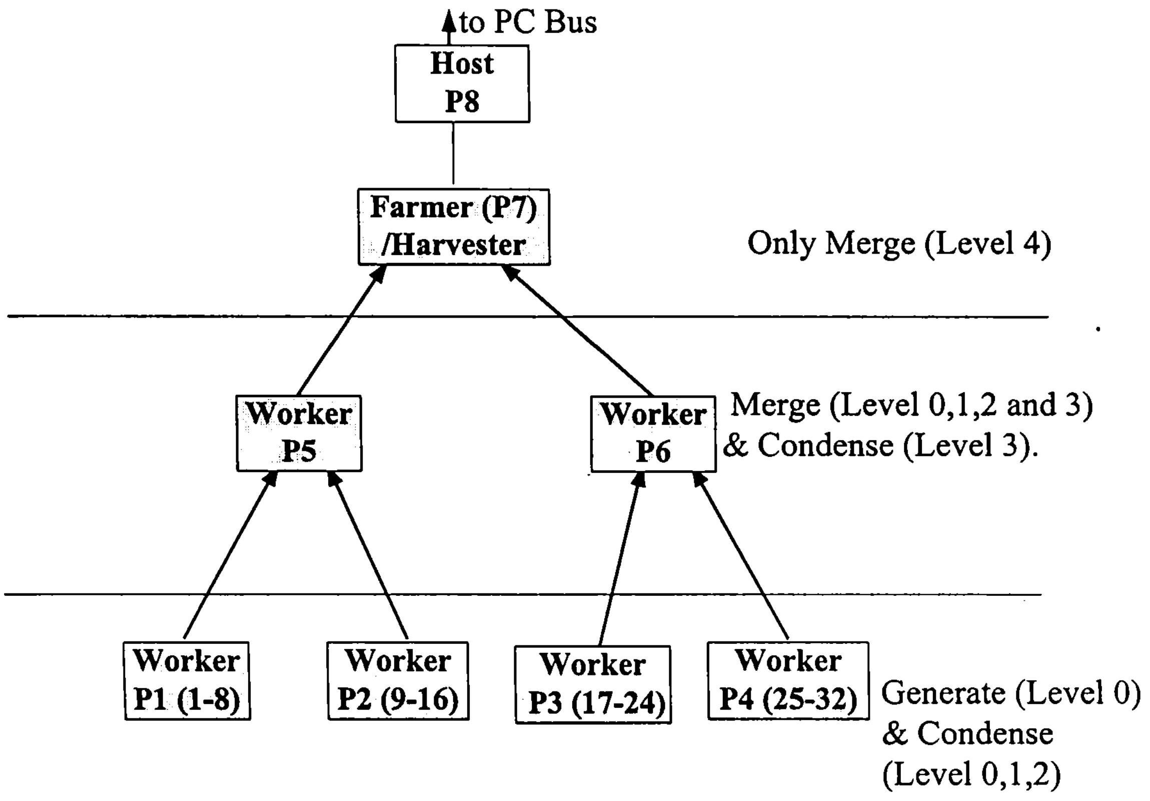
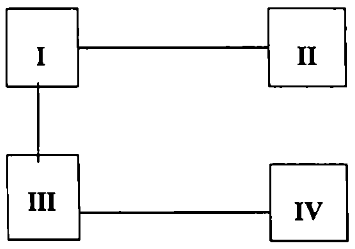


Fig.2.8 Transputer model of bottom-up algorithm with Balanced Tree topology

Each worker nodes, P1-P4, handles 8 planes each. The condensed o-codes from P1 and P2 communicated to P5 and similarly P3 and P4 to P6. Final result is harvested by the Farmer P7, after merging condensed o-codes from P5 and P6.



Processor I Farms the tasks to all the worker nodes II-IV and then process (share) the o-codes at the lower level, 0-3. The final result is merged, condensed and harvested by Processor I, the "Farmer".

Fig.2.9 Rectangular Mesh model on Transputer.

2.8 Time complexity

It may be noted that the time taken for the entire Octree generation shall be nearly ¼th of the sequential algorithm reported earlier [75Sojan97], with an extra amount of time being spent on the inter processor communication and final transfer of all sorted linear codes to the farmer.

Assume that the codes at each level are exponentially distributed.

$$\text{i.e.} \quad b_i = B\lambda^{i+1} \text{ ; where } \sum_{i=0}^{n-1} b_i = B \text{ and } 0 \le \lambda \le 1$$

There fore the time taken for the inter processor communication to generate the level 4 codes (for a 32x32x32 grid) estimated for the rectangular mesh model of Fig.2.9 is stated below.

$$\tfrac{1}{4}[b_3 + b_4 * 8] = (B/4) * [\lambda^4 + 8\lambda^5] = (B\lambda^4/4) * [1+8\lambda] \cong B/16 \quad \text{if } \lambda = \tfrac{1}{2}$$

This shows that the communication overhead is not substantial, if we agree to keep most of the generated codes (i.e. upto level 3) in each processor. Generalization to a larger number of processors (for a larger size grid) is straight forward, similarly considering the communication overheads for generating codes at higher levels. Of course the final transfer to the farmer is O(B), when all the codes will be available in processor I. The final merging of all the levels are done by the farmer after harvesting the results and thus the storage complexity per processor is O(B).

	Step I		Step II		Step III		Step IV		Step V	
Processor	**Comput-ation**	**Communication**	**Comput-ation**	**Communication**	**Comput-ation**	**Communication**	**Comput-ation**	**Communication**	**Comput-ation**	**Communication**
I	Generate Codes of level 0,1,2,3		Produce codes of level 4 in Proc.I		Merge codes of level 3,4		Merge codes of level 0,1,2		Merge codes of level 0,1,2 and sort the entire list.	
II	Generate Codes of level 0,1,2,3	Codes of level 3 moves to Proc. I	Merge codes of level 3			Codes of level 0,1,2 move to Proc.I				
III	Generate Codes of level 0,1,2,3		*Produce codes of level 4 in Proc.III *Merge codes of level 3	Codes of Level 3,4 moves to Proc.I from Proc. III			Merge codes of level 0,1,2	Codes of level 0,1,2 move to Proc.I		
IV	Generate Codes of level 0,1,2,3	Codes of level 3 moves to Processor III				codes of level 0,1,2 move to Proc.III				

Fig.2.10 Progressive generation of codes of different levels in each Processor.

2.9 Space complexity

However, during computation, the storage increases due to the requirement of holding the condensable codes. This extra storage would not be greater than (3/4B) for a 32x32x32 grid.

We have demonstrated the parallel implementation of the Octree Generation algorithm from CAT scan slices. The overheads in implementing on a Transputer [60Sojan'98] model of 4 nodes has been estimated. While the parallisation gives nearly a four-fold improvement in the execution time as expected, the communication overheads are not substantial.

2.10 Conclusion

We have demonstrated an algorithm to build octrees from CAT scan slices. The implementation of algorithm under concurrency and parallelism are also addressed in detail. However the o-code represents an average value of the density matter in the cubical space. Display of such an octree would lead to a "blocky" solid. In the Chapter to follow we address the issue of how the information contained within the cubical space can be encoded, with reduced loss, during the generation of octree from CAT scan slices.

Chapter 3

Generation of Octrees from raster scan with reduced information loss

Abstract

*Having addressed the issue of generating octrees from raster scan efficiently using parallel computation, it is worthwhile to ensure that the encoding is accomplished without much loss of information. Knowing that each cube usually stores an average of the intensity value in the cubical subspace, the encoding could lead to a "blocky" (as against "patchy") solid during regeneration. Retaining the entire intensity distribution in three dimensions may defeat the whole purpose of encoding. The present Chapter addresses the technique for octree encoding with minimum information loss. After considering the techniques based on vector quantization [183Gary84] [152Nasrabadi88] an efficient and easy to implement approach based on wavelet transform [184*Gross96*] is introduced.*

Often the reconstructed data from octrees results in a blocking effect because of the average density information assumed in the cubical space represented by the cubes defining the solid. The average density along with the variance in the cubical space also does not improve the situation. Recently, Wavelet transforms have been shown to be efficient in giving a compressed representation of square and cubical region [184Gross96] and the inverse wavelet transform, reconstructs the cubical space without much loss of information. The cubical region, which is represented by an octree leaf node along with the wavelet transform of the cubical region, can be thus stored in a very compact way. During reconstruction, the intensity distribution in the cubical space, as decided by the octree leaf node, can be recovered. It is shown that the wavelet transform can be progressively constructed during the raster to octree conversion [60Sojan98]. Starting at the voxel level, the 8 cubical blocks, from a pair of adjacent planes, are progressively combined depending upon the homogeneity of the density distribution among the eight voxels (condensation). During the condensation of the voxels (larger sized cubes at higher levels), the wavelet transform of the cubical space spanned by the eight voxels (or eight larger sized cubes) can also be computed and combined.

3.1 Reconstruction of solids from octrees

The common problem faced during the reconstruction process is to assign the intensity distribution to all voxels in the solid space, as the octree representation usually stores the average intensity in the cubical subspace. When cubes represented by a leaf element are large, the reconstructed solid becomes "patchy". Vector quantization can improve the reconstruction, wherein we attach a certain number of code words along with each cube. During reconstruction, the code words are replaced by the actual vectors, selected using code words [185Amir96]. In this Chapter, we propose that the wavelet transform can be used efficiently to encode the solid space. By attaching a certain amount of detail along with the approximation computed at some level, a good amount of reduction in the cubical sub-space is also achieved. We also demonstrate that the wavelet transform can be easily computed during the raster to octree construction, using a class of bi-orthogonal low pass and high pass filters. During reconstruction of the solid from the octree, the intensity distribution in the cubical sub-space encoded by the o-code is regenerated from the stored wavelet transforms.

3.2 Vector Quantization

A vector quantizer is a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over storage in a digital channel. The goal of such a system is data compression: to reduce the bit rate so as to minimize communication channel capacity or digital storage memory requirements while maintaining the necessary fidelity of the data.

Vector Quantization (VQ) is an effective data compression technique for speech and images where high compression ratio is desired. An important problem in VQ is how to design a codebook that is good for a source [186Gersho92] [187Linde80], and another important issue naturally associated with the problem is testing designed codebooks. In testing a designed codebook, we may consider two aspects: the optimality for a given source and the robustness for various other sources [188Neuhoof87]. The former has to do with good design of a codebook for a particular source, while the latter implies goodness of a designed codebook for other sources that cannot be targeted in the design process. Experimental observations on robustness of codebooks are introduced under the title of the quantizer mismatch problems in [189Gray75] [190Mauer79] [191Yamada84]. However, most codebook design problems are concerned with a faithful design of a codebook for a given source [186Gersho92]. Thus it is important in judging a codebook to validate its optimality for a particular source.

Kim et al [192Kim97] discusses a criterion for testing a vector quantizer codebook that is obtained by "training". VQ codebook is designed by a clustering algorithm using a training-set-distortion (TSD). The algorithm stops when TSD significantly decrease. In order to test the resultant codebook, validating-set-distortion (VSD) is calculated on a separate validating set (VS). Codebooks that yield small difference between the TSD and the VSD are regarded as good ones. However, the difference in VSD-TSD is not necessarily a desirable criterion for testing a trained codebook unless certain conditions are satisfied. A condition that is previously assumed to be important is that the VS has to be quite large to well approximate the source of distribution. This condition implies greater computational burden of testing a codebook. They discussed the condition under which the difference VSD-TSD is a meaningful codebook-testing criterion. Then, convergence properties of the VSD, a time-average quantity, are investigated. Finally they showed that for larger codebooks, a VS size as small as the size of the codebook is sufficient to evaluate the VSD. Kim et al consequently presents a simple method to test trained codebooks for VQ's.

It is worthwhile to note that the codebook gives a code representing a data distribution. But the dataset itself will have to be stored somewhere to be used for filling in the solid space during reconstruction. On the other hand, techniques based on wavelet transform facilitate the representation of the density distribution in a cubical space with a smaller dataset (called approximation and detail) from which the data distribution in the cubical space can be easily reconstructed during decoding. We therefore propose an approach to incorporate the wavelet compression, which combines elegantly with the synthesis of octrees from CAT scan slices.

3.3 Wavelet transforms

Wavelets are a mathematical tool for hierarchically decomposing functions. They allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the levels of detail present. Authors [193Stollnitz95a] [194Stollnitz95b] were intended to provide people working in computer graphics with some intuition for what wavelets are, as well as to present the mathematical foundations necessary for studying and using them.

Although wavelets have their roots in approximation theory [195Daubechies88], and signal processing [196Mallat89], they have recently been applied to many problems in computer graphics. These graphical applications include image editing [197Berman94], image compression [198DeVore92], and image querying [199Jacobs95]; automatic level of detail control for editing and rendering curves and surfaces [200Finkelstein94] [201Gortler95] [202Lounsbery93], surface reconstruction from contours [203Meyers94], and fast method for solving simulation problems in animation [204Liu94], and global illumination [205Christensen95] [206Christensen94] [207Gortler93] [208Schroder93] .

The Haar basis is the simplest wavelet basis. One-dimensional Haar wavelet is illustrated as follows.

3.3.1 The one-dimensional Haar wavelet Transform.

Suppose we are given a one-dimensional "image" with a resolution of four pixels, having the values

9 7 3 5

We can represent this image in the Haar basis by computing a wavelet transform. To do this, first average the pixels together, pair-wise, to get the new lower resolution image with pixel values

8 4

Clearly some information has been lost in this averaging process. To recover the original four pixel values from the two averaged values, we need to store some detail coefficients, which capture the missing information. In the above example, we will choose 1 for the first detail coefficient, since the average we computed is 1 less than 9 and 1 more than 7. This single number allows us to recover the first two pixels of our original four-pixel image. Similarly the second detail coefficient is -1, since 4 + (-1) = 3 and 4 - (-1) = 5.

Resolution	Averages	Detail coefficients
4	9 7 3 5	
2	8 4	1 -1
1	6	2

Table 3.1 Decomposition of four-pixel image.

Thus, the original image has decomposed into a lower resolution (two-pixel) version and a pair of detail coefficients. Repeating this process recursively on the averages gives the full decomposition. Finally, we will define the wavelet transform (also called the wavelet decomposition) of the original four pixel image to be the single coefficient representing the overall average of the original image, followed by the detail coefficients in order of increasing resolution. Thus, for one-dimensional Haar basis, the wavelet transform of our original four-pixel image is given by

6 2 1 -1

Storing the image wavelet transform, rather than the image itself, have a number of advantages. One advantage of the wavelet transform is that often a large number of the detail coefficients turn out to be very small in magnitude, as in the example of Table 3.1 above. Truncating, or removing, these small coefficients from the representation introduce only small errors in the reconstructed image, giving a form of "lossy" image compression. More details of this can be referred in [193Stollnitz95a].

3.3.2 One-Dimensional Haar wavelet basic functions

We have shown that one-dimensional images can be treated as sequence of coefficients. Alternatively, we can think of images as piece wise-constant functions on the half-open interval [0,1). To do so, we will use the concept of vector space from linear algebra. A one-pixel image is just a function that is constant over the entire interval [0,1). We'll let V^o be the vector space of all these functions. A two-pixel image has two constant pieces over the intervals [0, 1/2) and [1/2,1). We'll call the space containing all these functions V^1. If we continue in this manner, the space V^j will include all piece wise - constant functions defined on the interval [0,1) with constant pieces over each of 2^j equal subintervals.

We can now think of every one-dimensional image with 2^j pixels as an element, or vector, in V^j. Note that because these vectors are all functions defined on the unit interval, every vector in V^j is also contained in V^{j+1}.

3.3.3 Two-dimensional Haar wavelet transforms

There are two common ways in which wavelets can be used to transform the pixel values within an image. Each of these transformations is a two-dimensional generalization of the one-dimensional wavelet transform described in section 3.3.2

The first transform is called the standard decomposition [209Beylkin91]. To obtain the standard decomposition of an image, we first apply the one-dimensional wavelet transform to each row of pixel values. This operation gives us an average value along with detail coefficients for each row. Next, we treat these transformed rows as if they were themselves an image and apply the one-dimensional transform to each column. The resulting values are all detail coefficients except for single overall average coefficients. An algorithm to compute the standard decomposition is given by [210Stollnitz96]

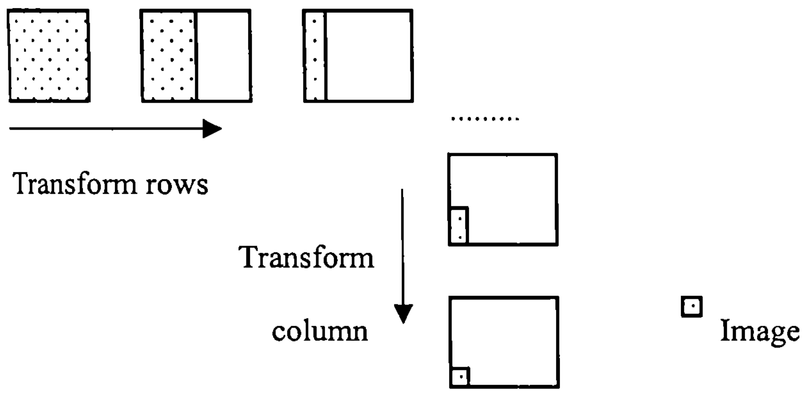


Fig. 3.1 Standard decomposition of image.

The second type of two-dimensional wavelet transform, called the non-standard decomposition [same as standard decomposition], alternates between operations on rows and columns. First, we perform one step of horizontal pair-wise averaging and differentiating to each column of the result. To complete the transformation, we repeat this process recursively only on the quadrant containing averages in both directions. Fig 3.2 shows all the steps involved in the non-standard decomposition.

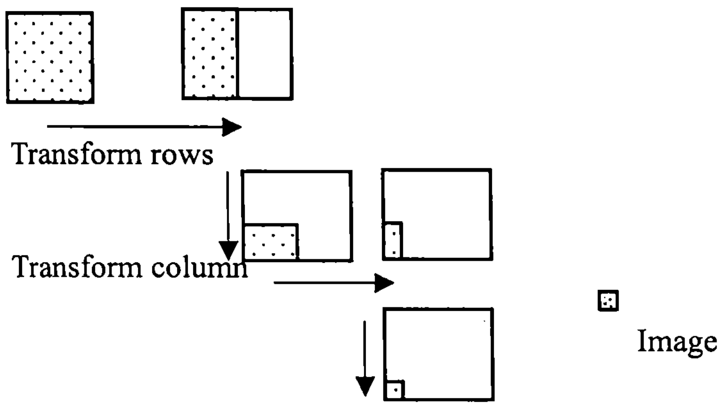


Fig.3.2 Non-standard decomposition of image

Bi-orthogonal Discrete Wavelet Transforms are computed by applying 1-D symmetric low pass and high pass filters on the given image to generate the signal approximation and detail [21]Thomas96]. Both the filter outputs are decimated by a factor of two. Thus the decomposition is space preserving, meaning that total number of points at the output of the filtering is equal to the number of points in the input. The inverse transform is calculated by up-sampling the transform by adding zeroes, applying the inverse filters and then combining the filter outputs. Fig.3.5 shows the computational scheme.

We have used the Haar wavelets for implementing the wavelet transform.

We use two type of filters viz.

(i) Low pass [$1/\sqrt{2}$, $1/\sqrt{2}$] and

(ii) High pass [$1/\sqrt{2}$, $-1/\sqrt{2}$]

The resulting octave decomposition in three dimension results in cubical subspaces of lower dimension computed recursively. As given by Muraki [212Muraki93], we compute the wavelet transforms along a row, then along the column and then along depth, using QMF filters mentioned above.

3.4 Computation of wavelet transform for the o-codes

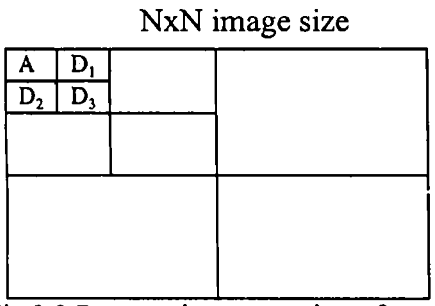


Fig.3.3 Progressive generation of approximation.

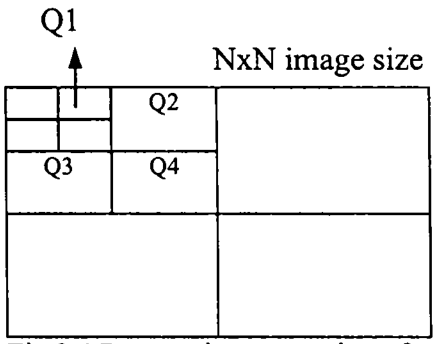
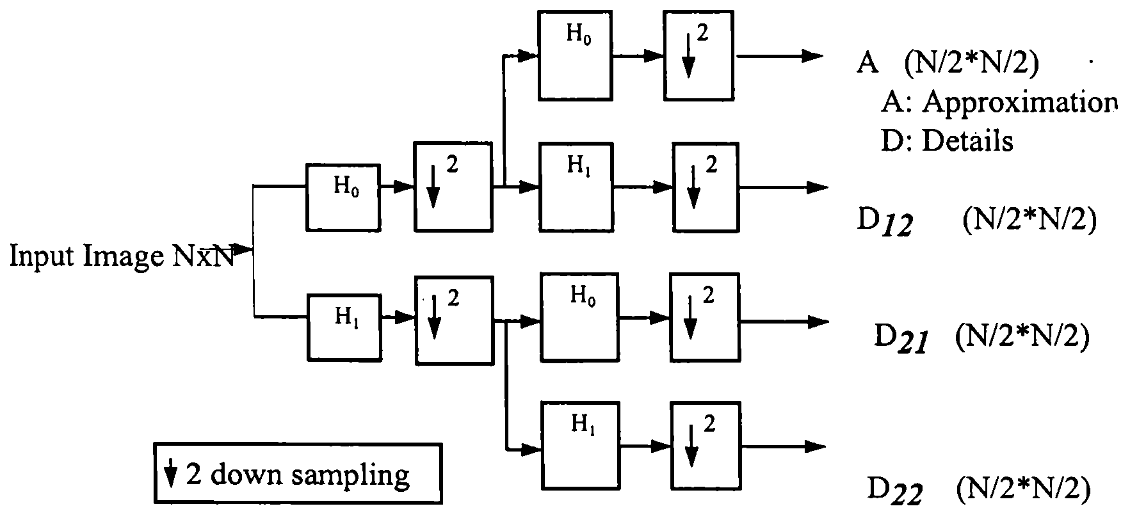


Fig.3.4 Progressive generation of q-code at level 1.

Fig.3.3 shows the progressive generation of the approximation and details at resolution (N/4 x N/4), from an image of size NxN. Assuming that the details at higher resolution can be neglected, one can reconstruct the original image (with certain amount of loss) from the approximation (A) and detail (D_1,D_2,D_3) using the inverse wavelet transform. On the other hand, look at the octree construction from raster. For ease of explanation we illustrate the process of generating quadtrees from raster image. Four pixels positioned as in Fig.3.4 are combined to form a q-code Q1 at level 1 (representing a size of 2x2). Let us attach the approximation and details from the four pixels along with Q1. Similarly q-codes Q2, Q3 and Q4 shall be generated in the process of looking for larger homogeneous spaces. As Q1, Q2, Q3 and Q4 *condense* to produce a q-code of size 4x4, the approximation and the details computed along with Q1, Q2, Q3 and Q4 are used to find the approximation and details at resolution 2x2. (We neglect the detail and use only the approximation.) Now we have a q-code representing a size of 4x4 along with approximation and details corresponding to a size of 2x2. The idea of computing the approximation and details, to q-codes representing larger size can be easily extended. The idea of condensation to q-codes representing larger square can also be easily extended to cubical blocks. The wavelet transform is computed on 8 voxels (at level 0), 4 each from adjacent planes. As the o-codes corresponding to cubes of larger size are generated, the approximation and details are computed recursively.

The computation of wavelet transform thus progresses from pixels (as shown in Fig.3.5) and produces the approximation and details for each condensation. Corresponding to some lower level of resolution, we propose to retain the details also. It is then possible to reconstruct the intensity distribution in the cubical space by suitably interpolating the details [184Gross96].

The explanation given above applies equally well to cubical blocks generated during the synthesis of octrees. The approximation and details are constructed by considering x,y,z directions. Finally each o-code shall have a set of approximations



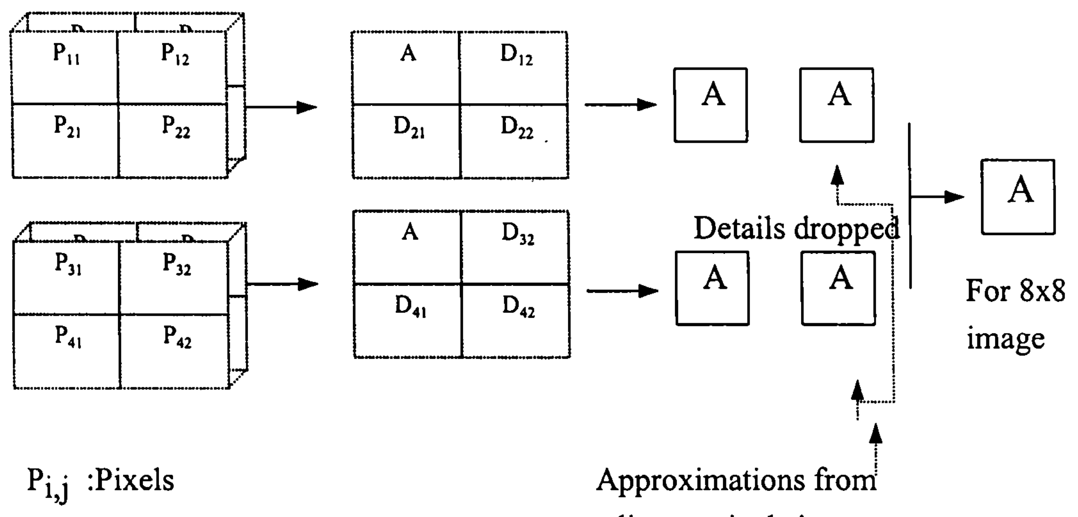

P_{11} P_{12} P_{21} P_{22}
P_{31} P_{32} P_{41} P_{42}
A D_{12} D_{21} D_{22}
A D_{32} D_{41} D_{42}
A
Details dropped
For 8x8 image
$P_{i,j}$:Pixels
Approximations from adjacent pixels in same line.

Fig.3.5 Computation of Wavelet Transform for a 2D image

and details (depending upon where we stop the computation of approximations). The increase in storage space is thus decided by the level to which the approximations and details are to be computed.

3.5 Reconstruction of the solid

As in the case of octree decoding each cubical block is positioned depending upon the co-ordinate information contained in the o-code. The corresponding cubical space is filled using the density distribution constructed from the approximation and detail. Fig.3.6 shows the scheme of inverse wavelet transform [193Stollnitz95] resulting in the generation of data distribution. Depending upon the details dropped out during the computation of wavelet transform, the generated dataset from inverse wavelet transform may have to be interpolated.

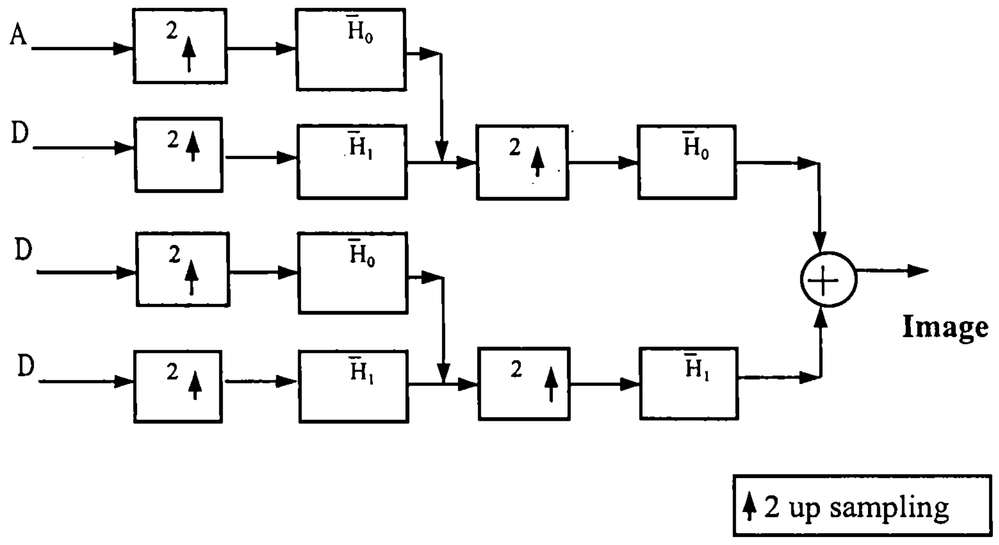


Fig.3.6 Computation of inverse Wavelet Transform

3.6 Conclusion

This Chapter summarizes how the wavelet transform can be computed during the condensation process encountered in the raster to octree conversion. The wavelet transform corresponding to a certain amount of resolution along with some detail is stored for every o-code. During re-construction, the o-code specifies the size and position of the cubical space, while the wavelet transform helps to reconstruct the intensity distribution.

Chapter 4

Display of objects represented by Octree

Abstract

Several algorithms help to visualize medical gray level volumes. For clinical applications it is vital that the generated images represent reality. Volume rendering is a technique for visualizing sampled functions of three spatial dimensions by computing 2D projections of a colored semitransparent volume.

Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) are the medical imaging modalities that deliver cross-sectional images of the human body. If we record sequences of adjacent images, we have a 3D volume representation of that part of the anatomy under consideration. Unlike 2D imaging, the presentation of gray-level volumes is subject to a number of choices. As those objects present in the volume may obscure each other, one has to decide which object to visualize. Furthermore, one can choose the form of presentation, for example, transparent, as cut planes or as surfaces. If one uses a surface visualization, there are different ways of determining the surface, the surface normals from the gray-level volume, and the subsequent shading.

Volume rendering addresses the problem of displaying the 3D information on a 2D screen. Though the final display gives the outer surface of the solid profile in most of the rendering algorithm reported [^{119}Lorensen87] [^{216}Raj96], the rendering sequence posts the complete volume information on the display screen and by virtue of occlusion the surface alone shows up. We briefly review some of the algorithms reported for displaying 3D volume information. We have proposed an algorithm for displaying octree with transparency shading, Fig.4.5 and Fig.4.6.

4.1 Marching Cubes

The Marching Cubes (MC) algorithm is a commonly used method for generating isosurfaces. Rendering isosurfaces, as opposed to volume rendering, is an area of continued interest to the visualization community. Availability of dedicated hardware to speed up surface rendering and thus achieve real-time interaction has been a major contributing factor to this interest. Several algorithms can generate isosurfaces from 3D data. The Marching Cubes (MC) algorithm [213Fuchs77] [214Wyvill86] [119Lorensen87] has, by far, been the most popular one in generating high-quality surface representation. Experience with visualizing medical data sets has proved that, despite the attractiveness of the surface rendering approach, the number of surface primitives (triangles) generated by the MC algorithm can be prohibitive to achieve a reasonable rendering speed. To circumvent this problem, original datasets are commonly down sampled before applying the MC algorithm to yield a surface representation of reasonable size. Thus there is a trade-off between surface detail and rendering speed.

Several attempts have been made to reduce the number of triangles generated by the MC algorithm. The most notable, the decimation algorithm by [215Schroeder92], substitutes the triangular mesh obtained by the MC algorithm with a simple mesh generated from a subset of original vertices. Vertices are classified as one of the six types based on their interconnections with neighbouring vertices. Each vertex is a candidate for deletion and the error resulting from removing a vertex is evaluated. If the resulting error is within a user-specified limit, the vertex is deleted from the mesh. The algorithm has been shown to reduce the number of triangles by as much as 90% without much distortion. One major problem with this method is that a large number of triangles is generated only to be eliminated later on.

The MC algorithm also generates an excessively large number of triangles to represent an isosurface. Generating many triangles increases the rendering time, which is directly proportional to the number of triangles. Raj Shekar et al [216Raj 96] presents a decimation method to reduce the number of triangles generated by the MC algorithm. Decimation is carried out within the framework of the MC algorithm before creating a large number of triangles. Four major steps comprise the reported implementation of the algorithm :

a) Surface Tracking

b) Merging

c) Crack patching and

d) Triangulation.

Surface tracking is an enhanced implementation of the MC algorithm. Starting from a seed point, the surface tracker visits only those cells likely to compose part of the desired isosurface. This results in up to approximately 80% computational saving. The cells making up the extracted surfaces are stored in an octree that is further processed. A bottom-up approach as described in Chapter 2 is taken in merging the cells containing a relatively flat approximating surface. The finer surface details are maintained. Cells are merged as long as the error due to such an operation is within a user-specified error parameter, or a cell acquires more than one connected surface component in it. A simple, yet general, crack patching method is described that forces edges of smaller cells to tie along those of the larger neighbouring cells. Patching does not introduce new triangles. The overall saving in the number of triangles depends both on the specified error value and the nature of data. Raj [216Raj96] demonstrated savings of more than 90% for two artificial datasets and an MRI head dataset for an error value of less than half the minimum voxel dimension. Use of the hierarchical octree data structure also presents the potential of incremental representation of surfaces. Raj [216Raj96] generated a highly smoothed surface representation, which can be progressively refined as the user-specified error value is decreased.

Shu et al [217Shu95] report an adaptive MC algorithm. The MC algorithm is first applied to cells of a given size, which is a power of 2 (2x2x2, 4x4x4, 8x8x8 and so on). If the approximating surface is not flat enough based on a curvature criterion, the initial cell is subdivided. The process continues until either the approximating surface is satisfactory or the initial cell is divided into 1x1x1 cells. There are several draw backs to this approach. Primarily, savings of 55% do not compare favorably with those of nearly 90% offered by the decimation algorithm described above and the algorithm presented in this paper. Moreover, starting the process with cells of a fixed size may not fully exploit the possible reduction. Another drawback is that applying curvature criterion on the edges for splitting cells may cause loss of finer features present within the cells. In addition, crack patching replaces the cracks with equivalent polygons, which may counter the saving. And finally, the algorithm is limited to datasets of resolutions of the power 2.

Montani et al [218Montani94] have proposed a discretized MC algorithm where the edge intersections are approximated by edge midpoints. [216Raj96] argued that the error introduced with mid point selection is within acceptable limits. The saving in Montani's approach results primarily from the removal of smaller facets and the merging of the co-planner facets. Mid point selection improves the chances that facets from neighbouring cells will be coplanar. The investigations also report 80-90% savings in the number of triangles. The algorithm we propose can incorporate this idea and result in further saving.

Wilhelms and Van Gelder [219,220Wilhelms90,92] report use of an octree data structure to enhance the MC algorithm. Visiting only a subset of all the possible cells increases the efficiency. Octrees, with their hierarchical structure provide a natural framework for avoiding the unnecessary cells. The enhancement in the [216Raj96] work is however, achieved through tracking surface.

The Marching Cubes (MC) [214Wyvil86] [119Lorensen87] method demonstrated that isosurface extraction can be reduced to solving a local triangulation problem through a table lookup. To achieve this, the MC method visits each and every cell of the data. More sophisticated searching strategies visit practically only the cubes that contain the isosurface [221Cignoni96] [222Livnat96].

Datasets of several gigabytes are found in medicine as well as the geo-sciences. The size of isosurface extracted from these datasets can reach several million polygons, many of them are less than one pixel in size. The computation of all the local triangulations can be very time consuming and the huge number of polygons can easily overwhelm even the most powerful graphics accelerators, leading to poor interaction. One current approach to the large number of polygons problem is to apply mesh reduction techniques [216Raj96] [223Man96] to isosurface either as a post process to the extraction phase or during the extracting phase itself [224Poston97]. However, mesh reduction is expensive and requires extracting the entire isosurface for examination. Further more, a change in the isovalue requires the full extraction of a new isosurface and the re-application of the mesh reduction step.

A different approach is to employ ray-tracing techniques that do not need to create an intermediate polygonal representation. Ray-tracing, nevertheless, does not take advantage of graphics hardware and requires a large number of CPUs for interactivity [225Parker98].

Livnat and Hansen [226Livnat 98] propose a new approach, which further reduces the search, construction and display, to polygonal isosurface extraction that is based on extracting only the visible portion of the isosurface from a given view point. It was demonstrated that the reduction in the isosurface size can be substantial (93%) which makes it attractive for remote visualization. The visibility tests are performed in software to determine the visible cells. These tests are based on hierarchical tiles and shear-warp factorization. The second phase resolves the visible portions of the extracted triangles and is accomplished by the graphics hardware.

While the latest isosurface extraction methods have effectively eliminated the search phase bottleneck, the cost of constructing and rendering the isosurface remains high.

Many of today's large datasets contain very large and complex isosurfaces that can easily overwhelm even state-of-the-art graphics hardware. The proposed approach by Livnat and Hansen is output sensitive and is thus well suited for remote visualization applications where the extraction and rendering phases are done on a separate machine. The authors concluded with their plan in future optimization and parallelizing the code for a larger data sets.

4.2 Octree-Based Decimation Of Marching Cubes surfaces

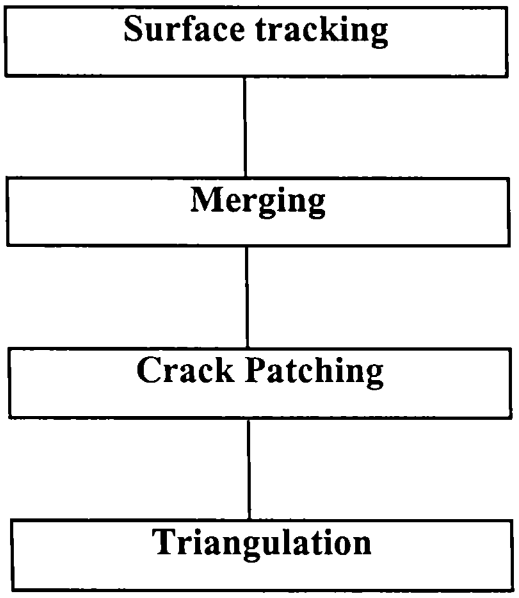


Fig. 4.1 The four steps in the reported implementation of the Octree-based decimation algorithm [216Raj96]

The octree-based decimation algorithm involves four steps, as outlined in the Fig.4.1 above. The first step is the surface tracking, which is the application of an enhanced MC algorithm to the dataset. Surface tracking identifies each cell through which the isosurface passes. The result of surface tracking is saved in an octree data structure. Several passes are made through this octree to achieve a compact surface representation. The octree is first processed to replace simple cells by a larger cell, if allowed by the merge criteria, which we referred as *condensation* in Chapter 2. This adaptive technique will develop cracks at the interface of cells with different resolutions. Such cracks are removed in a subsequent pass. The final step is to create the triangles that make up the surface. Each step of the algorithm is explained in detail by Raj [216Raj 96].

4.3 Parallel Display of Objects Represented by Linear Octree

Most existing serial algorithms that display 3D objects on a 2D screen are found to be too slow to process the large amount of volume data in a reasonable time. Hence one way to increase the performance of the display algorithm is to process individual volume data elements (voxels) in parallel. The first part of [227Ibaroudence95] presents a brief overview of the linear octree data structure [52Gargantini82] and the second part focuses on the parallel display of such objects. Ibaroudence and Raj [227Ibaroudence95] have shown that, for an object represented by linear octree and enclosed in a $2^n x 2^n x 2^n$ universe cube, the maximum number of voxels that can be processed in parallel is 3^n and the maximum number of time steps required to display such an object is 4^n. They present a set of formulae which identify the processing element (PE) as well as the time step in which a given linear octree node which must be processed by a given PE, at some specific time step, was presented, along with a strategy for determining whether a PE is active or idle.

To display a 3D object, represented by an octree, on a 2D screen, nodes are recursively processed in either a back-to-front (BTF) or a front-to-back traversal (FTB) sequence. In a BTF traversal sequence, nodes that are invisible to the viewer are visited first, followed by the nodes that have one, two, and three visible faces, in this order. At the conclusion of the display process, all the hidden surfaces will be painted over by the occluding nodes that entered the display pipeline at later times. Ibaroudence [227Ibaroudence95] assumed orthogonal projections are used to display 3D objects on a 2D screen.

The traversal order of the descendants of a given node in an octree is dependent on the viewer position, which is defined by the view plane normal, N. The traversal sequence of the octree nodes, which will eliminate all the hidden surfaces, can easily be inferred from the co-ordinates (a,b,c) of the unit view plane normal, N_U. The three binary variables x_0, x_1, and x_2 whose values are given by the following expressions.

$$x_2 = \left| \begin{array}{l} 0, \text{if } a \leq 0 \\ 1, \text{otherwise} \end{array} \right.$$

$$x_1 = \left| \begin{array}{l} 0, \text{if } b \leq 0 \\ 1, \text{otherwise} \end{array} \right.$$

$$x_0 = \left| \begin{array}{l} 0, \text{if } c \leq 0 \\ 1, \text{otherwise} \end{array} \right.$$

When viewing the descendants of a given node of a linear octree, the octant number of the closest sub-octant from the viewer is given by

$\Sigma_{i=0}^{2} x_i 2^i$ and its binary representation by $x_2\ x_1\ x_0$. Similarly, the octant number of the farthest sub-octant from the viewer is given by

$\Sigma_{i=0}^{2} x'_i 2^i$ and its binary representation by $x'_2\ x'_1\ x'_0$, where x'_i is the bit complement of x_i.

The back-to-front processing of the linear octree nodes can be summarized by the following four steps.

1. Process the farthest sub-octant from the viewer. Its locational number is $x'_2\ x'_1\ x'_0$ when written in binary form.
2. Process the three face neighbours of the farthest sub-octant from the viewer. Their octant numbers are $x_2\ x'_1\ x'_0$., $x'_2\ x_1\ x'_0$, $x'_2\ x'_1\ x_0$. These octants can be processed in parallel.
3. Process the three face neighbours of the closet sub-octant from the viewer. Their octant numbers are $x'_2\ x_1\ x_0$, $x_2\ x'_1\ x_0$, $x_2\ x_1\ x'_0$. These three octants can be processed in parallel.
4. Process the closest sub-octant from the viewer. Its locational number is $x_2\ x_1\ x_0$.

It is important to note that the octree nodes processed in later steps are never occluded by any previous step in the above four-step procedure.

Ibaroudence [227Ibaroudence95] have treated the nodes of the linear octree [52Gargantini82] as a set of voxels, i.e. each node is split up to the voxel level.

Hence, in terms of space efficiency, their representation of 3D objects using a non-condensed linear octree is no better than the cuberille [91Herman and liu79]. It is important to note all the PE's are busy at all times. Hence, PE utilization is less than optimal. However, using the organization described in their paper, they were able to exploit the maximum parallelism in the display of a 3D object represented by linear octree. Furthermore, they deterministically identified the PE and the time step at which a given voxel is processed.

4.4 Transparency Shading

Marc Levoy [10Levoy90] addresses the problem of extending volume rendering to handle polygonally defined objects accommodating transparency shading.

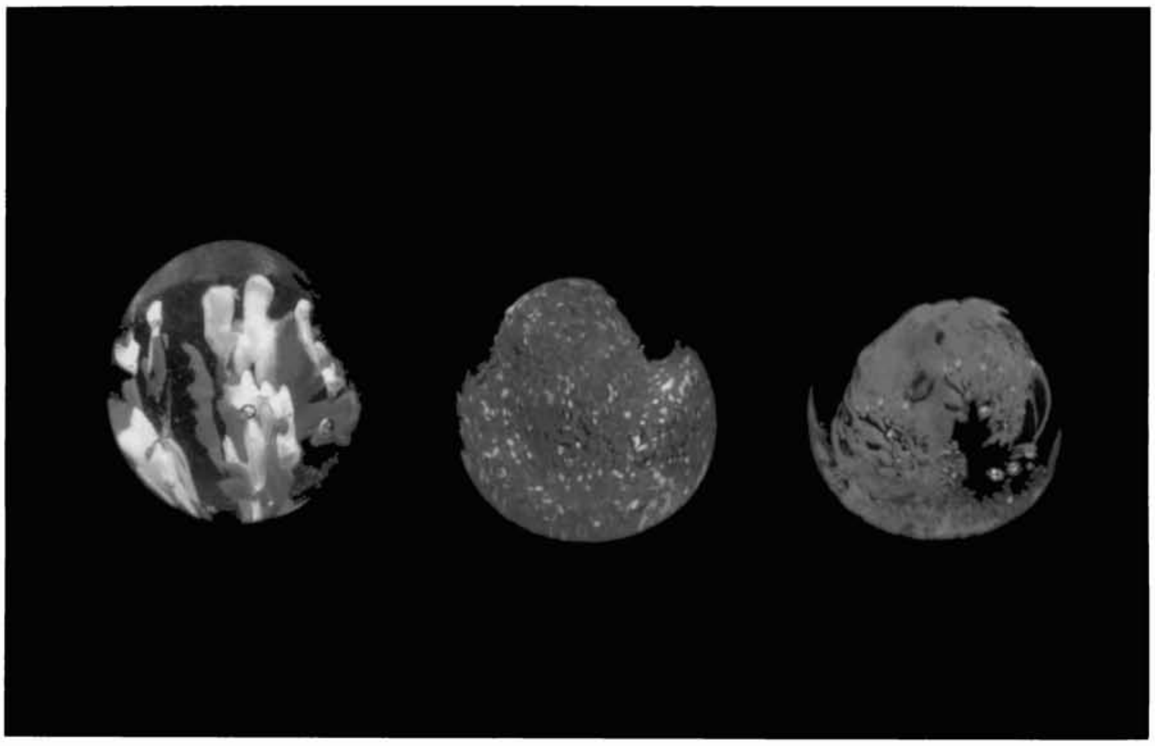


Fig. 4.2 Display of Transparency Shading

He proposes a hybrid ray-tracing algorithm. Rays are simultaneously cast through a set of polygons and a volume data array. Samples of each are drawn at equally spaced intervals along the rays, and the resulting colors and opacities are composited together in depth-sorted order. To avoid aliasing of polygonal edges at modest computational expense, he uses a form of selective super-sampling. To avoid errors in visibility at polygon-volume intersections, he gave special treatment to volume samples lying immediately in front of and behind polygons. Levoy evaluated the cost, image quality, and versatility of the algorithm using data from 3D medical imaging applications.

4.5 Display of Octree encoded Solids

Octree encoded solids are displayed by rendering each of the cubes corresponding to the linearly ordered o-codes. Since the linear ordering is in terms of the increasing order of z co-ordinates, the cubes (projected in a suitable direction) are displayed in a back-front order. As a natural consequence, the hidden faces get eliminated. Fig. 4.3 shows a typical octree display sequence.

The cubes are displayed in the (1,1,1) direction and the resulting 3 planes of the cubes are shaded by three different colors.

4.6 Display of Surface information from Octree

It may be observed that the interior details of the solid get occluded in the process of display, finally showing up the surface only. In order to speed up the rendering process, it would be worthwhile to extract the surface information alone and display

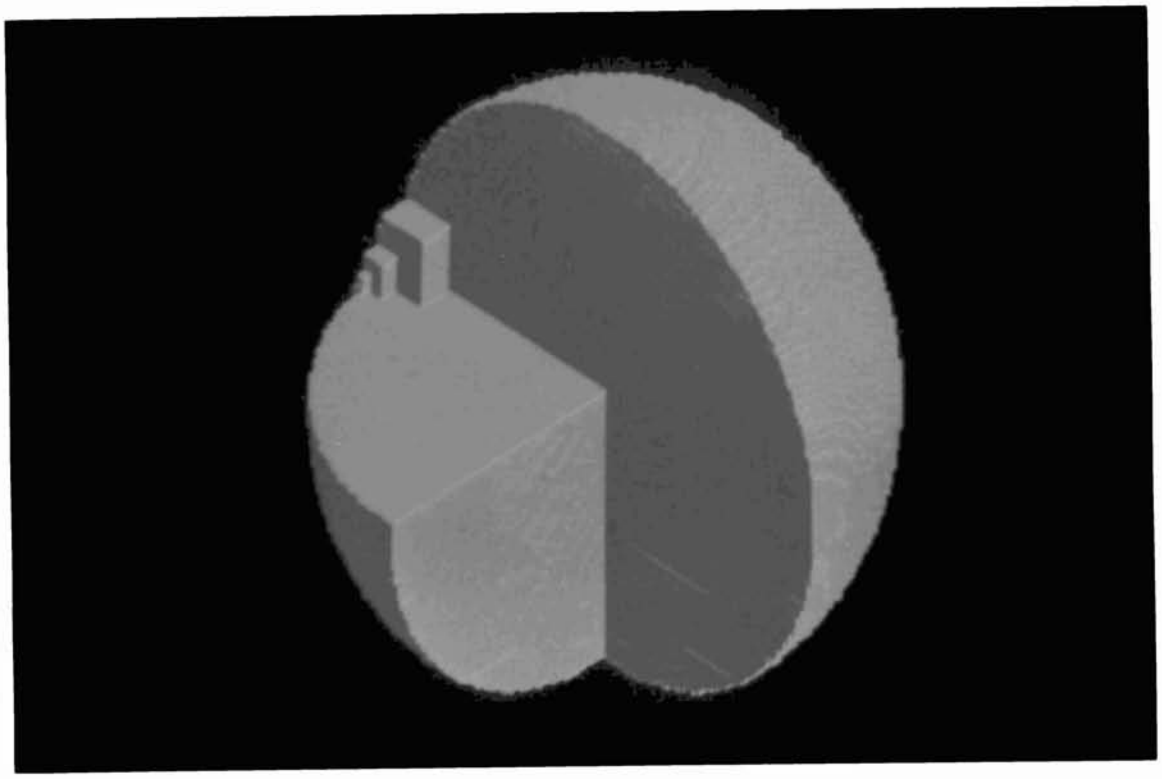


Fig. 4.3 Display of Octree encoded solids.

the same. The surface information is extracted by dropping all the cubes, which have neighbouring cubes all around. In order to extract the surface information, those cubes having neighbours in all the 6 directions (top, bottom, left, right, back, and front) are removed from the octree. The remaining set of o-codes represent the outer shell of the solid profile.

In order to speed up the process of extracting the surface, the octree is hierarchically ordered following [79Unni87]. Starting from the lowest level (i.e. corresponding to the pixel level)· the six neighbours of a given o-code, O, are calculated [52Gargantini82]. The o-codes corresponding to the neighbours are searched at the same level as well as the higher level looking for a match. If all the six neighbours are located in same or higher levels, the given o-code O is declared to be internal and removed. It may be noted that each level in the hierarchical order is linearly sorted, there by facilitating a binary search. Also noting an o-code of same level, the o-code for search in the higher levels is generated by removing the digits from the least significant side. Practical observation show that only o-codes at lower levels are left over in the process of elimination of internal o-codes. The left over o-codes are sorted in linear order and rendered.

4.7 Display of octree for Medical images.

Three-dimensional Medical information usually has variation in density and it is often desirable to show the internal matter also along with the outer solid profile. The normal octree rendering algorithms do not permit this because of the back-to-front rendering of the o-codes. We propose a method to display the inner details also along with the solid profile.

We assume hierarchical ordered octree. For each o-code visited at any level k ($0 \leq k \leq n$), its neighbouring o-code is computed in the viewing direction. (We assume viewing direction to be any of the 26 directions around the cube as shown in Fig. 4.4) The o-codes corresponding to the neighbour, O' is searched at the same or higher level of hierarchy. We then check if the neighbour is in front of or behind o-code O in the viewing direction. The rendering algorithm shall be invoked for level 0 to n-1 as shown below in Fig.4.5

Ref. Fig.4.6 the *NEIGHBOUR* function returns a neighbour *o* of same or higher level for an o-code o_k at level k. The function also ensures that the neighbour is inside the

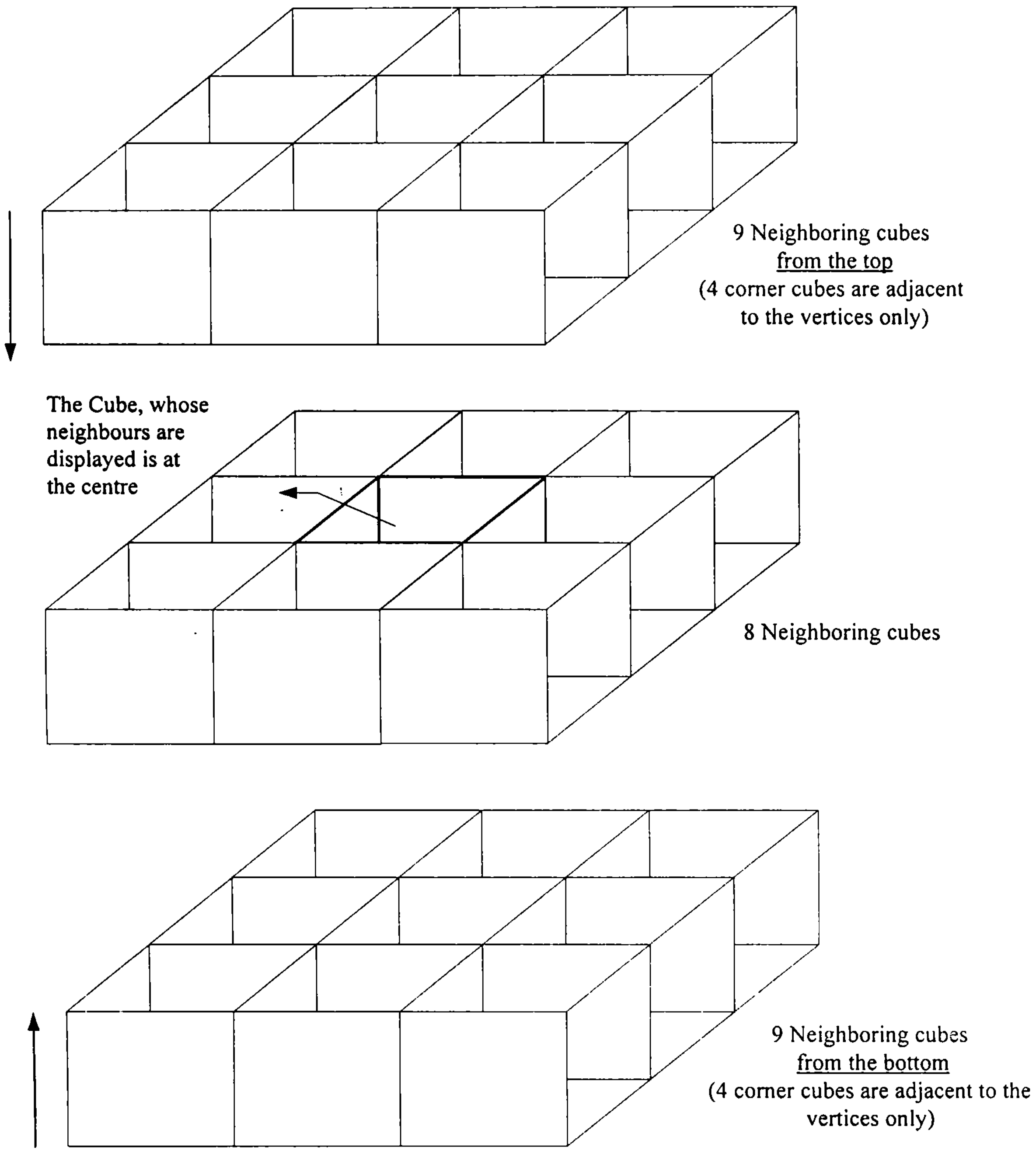


Fig. 4.4 Viewing Directions of a cube.

A total of 9+8+9 = 26 neighbouring cubes are shown in dotted lines.
8 Neighboring cubes, four from each plane, are adjacent to the respective vertices of the cube only.

solid profile and returns a positive number on success. The recursive function *DISPLAY(o_k,o)* renders the denser material inside as seen through. As shown in Fig.4.6 consider o_k is closer to the observer than o. (The function ρ(o) and level(o) returns the average density and the level of the computed neighbour). If $\rho(o) < \rho(o_k)$, Fig.4.6 computes more neighbours in the same direction removing the computed neighbours from the octree. There after the o-code is displayed. As the recursion works the codes are displayed in the increasing order of density with the densest code displayed last. However if the $\rho(o) \geq \rho(o_k)$ the ρ(o) is updated with the $\rho(o_k)$ there by achieving the integration of density in the direction of projection. The less denser o-code o_k is first rendered before the neighbours in the same direction are rendered. It may be noted that each rendered neighbour carries forth a portion of the densities of the inner matter. Though we have suggested straight summation of density, one could go in for an exponential averaging as suggested in [10Levoy90]. The case corresponding to o_k being further to the observer than o, can similarly be explained.

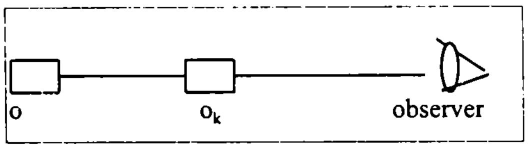
```
for level k= 0 to n-1 do
    for any o-code $o_k$ at level k
      if not marked ($o_k$)
          if neighbour($o_k$,d) then
                display ($o_k$,neighbour($o_k$,d));
```

ig. 4.5 Algorithm to invoke the rendering algorithm of Fig.4.6

rked nodes will not be considered again during the traversal of the octree.

```
lisplay($o_k$, o)

'/A denser material inside should be seen through //
:ase 1:
    // $o_k$ is closer to observer than o //
    $\rho(o) < \rho(o_k)$ :
    {
        mark(o);
        if neighbour(o,d) then
            display(o, neighbour(o,d);D($o_k$));
    }
```



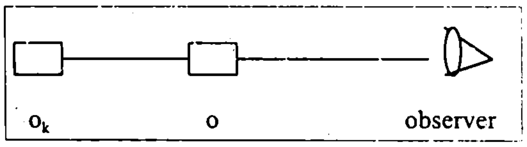
```
    $\rho(o) \geq \rho(o_k)$ :
    {
        D($o_k$);
        $\rho(o) \leftarrow \rho(o) + \rho(o_k)$;
        mark(o);
        if neighbour(o,d) then
            display(o, neighbour(o,d));
        else
            D(o);
    }
```

mark(o): Nodes are marked during computation and are not visited again.

ρ(o) Returns the density of 'o'.

neighbour(o,d) returns a neighbour of larger size in direction 'd'. If the neighbour doesn't exist, it returns a negative value.

```
case 2:
    // $o_k$ is further to observer than o //
    $\rho(o) \leq \rho(o_k)$ :
    {
        D($o_k$);
        $\rho(o) \leftarrow \rho(o) + \rho(o_k)$;
        mark(o);
        if neighbour(o,d) then
            display(o, neighbour(o,d));
        else
            D(o);
    }
```



Display the individual cubes in the direction d.

```
    $\rho(o) > \rho(o_k)$ :
    {
        D($o_k$);
        mark(o);
        if neighbour(o,d) then
            display(o, neighbour(o,d))
    }
```

Fig. 4.6 Rendering algorithm for bringing out denser material inside the solid profile

e algorithm in Fig. 4.5 visits each node only once and does not invite neighbour mputation a second time for a given o-code. The algorithm therefore executes in ne proportional to the number of o-codes, though the integration could take an extra nount of time.

may be noted that the algorithm accumulates the internal details in the cubes lying ternally. Therefore o-codes corresponding to internal cubes bring out the internal atter when projected finally. Here again the hierarchical ordering helps to reduce e search complexity corresponding to the computed neighbour. We assume andard functions that operate on o-codes to decide:

a) on computation of neighbours.

b) if one o-code is ahead of the other and

c) if a given o-code has reached the boundary of the solid

is worthwhile to note that an o-code visited at a higher level (during the search for eighbours) need not be considered again. So the list of o-codes at higher level is rogressively pruned down.

.8 Conclusion

Ve have discussed the algorithm for displaying octree encoded solid. We have also roposed an algorithm for displaying octree with transparency shading. It is strongly elt that the accommodation of transparency in displaying is most essential in medical naging.

Chapter 5

Conclusion

While attempting to build a practical system for processing 3D data, many issues like time and space required for implementing algorithm demands attention. Though the 3D data could be generated using mathematical techniques as in CAD and simulation, the practical applicability of a space efficient approach, is suggested as a viable approach. The octree then comes up as a natural candidate for representing CAT scan slices of medical images. We chose to address the related problems in using the octrees under three main headings.

1. Generation of octree from CAT scan slices
2. Reducing the information loss during the formation of octrees and
3. Rendering the stored information

The 3D data available in Medical imaging are often available in layers or slices. Any well-conceived 3D imaging system based on octree should therefore first address .the issue of building octree from the slices. We have addressed these problems by considering two adjacent planes at a time, thereby resulting in good space efficiency, during condensation [P1]. Improvement in speed of execution of the octree generation algorithm is also explored using parallel processing techniques. Needless to say, any parallel algorithm should assume a basic interconnection of processing elements and this has motivated us to propose an algorithm based on Transputer [P2] for-high-speed implementation of octree generation.

Noting that data contained in the cubical sub-space of the octree can not be always equal to an average value, we have used the wavelet transform technique to handle, the non-homogeneous data set, arising in medical imaging, within a cubical subspace. It is demonstrated that the octree generation algorithm can be modified to efficiently accommodate the wavelet compression up to a desired level. Though we have explored the possibility of using Vector Quantization (VQ), the relative difficulty in handling VQ technique along with the conversion dissuaded us from incorporating the same during the octree generation.

Since the processing algorithm using octree are plenty in literature, we did not venture into proposing any new approach in this area. However, we have shown how the information represented as octree can be rendered, so as to accommodate the transparency and opacity in the 3D data.

In Chapter 1 we introduced the octree and addressed the complete nature of problems encountered, while building an imaging system based on octrees. An efficient Bottom-up recursive algorithm and its iterative counterpart for the raster to octree conversion of CAT scan slices were discussed in Chapter 2. In an attempt to further improve the speed of generating the octree from the slices, the possibility of utilizing the inherent parallelism in the conversion program is also explored in Chapter 3. The implementation of the proposed parallel algorithm on a set of interconnected Transputer is further explored in the same Chapter. The octree node, which stores the volume information in a cube often stores the average density information over the cubical subspace. The storage of average density information could lead to "patchy" distribution of density during the image reconstruction. In an attempt to alleviate this problem, we have explored the possibility of using VQ to represent the information contained within a cube. Considering the ease of accommodating the process of compressing the information during the generation of octrees from CAT scan slices, we have proposed the use of wavelet transforms to generate the compressed information [P3] in a cube. The modified algorithm for generating octrees from the slices is shown to accommodate the wavelet compression easily. Rendering the stored information in the form of octree is a complex task, necessarily because of the requirement to display the volumetric information. In Chapter 4 we have demonstrated that the material inside the solid can be brought out using the transparent shading. In other words, the rays traced from each cube in the octree, sum up the density en-route, accounting for the opacities and transparencies produced due to variations in density. Finally this Chapter sums up the contributions in the thesis, giving guidelines for further work.

5.1 Further Work

We have noticed that there could be a large similarity of regions inside the solid profile either at the same level or at a different level of the generated medical images from CAT scan slices. This will be in the same direction or in a transformed state, leading to further compression of data, using fractals.

It is worth experimenting the proposed Transputer model [60Sojan98] on a network of nodes for processing the CAT scan slices, to have a real time visualization and analysis. The proposed display techniques in Chapter 4 for bringing out the inner details for the synthesized solid, are vital for Computer Aided Surgery and diagnosis.

[1] David F.Rogers, "Procedural Elements for Computer Graphics", Pub. by Mc Graw Hill International Editions, 1988.

[2] William Newman, Robert Sproull, 2 Edn., Pub. by Tata Mc Graw Hill International, 1997.

[3] Michael L Rhodes, "Computer Graphics in Medicine", IEEE CG&A, Vol.10, No.2, March 1990, pp20-23.

[4] R.A.Derbin et al, "Volume rendering", Computer Graphics, Proc. SIGGRAPH Vol.22, No. 4, Aug 1988, pp 65-74.

[5] T.Porter et al, "Compositing Digital Images", Computer Graphics, Proc. SIGGRAPH Vol.18, No.3, July 1984, pp 253-260.

[6] Ulf Tiede et al, "Investigation of Medical 3D-Rendering Algorithms", Vol.10, No.2, March 1990, pp 41-53.

[7] Linda J Brewster et al, "Interactive Surgical Planning", IEEE CG & A, Vol.4 , No.3 , 1984, pp 31-40.

[8] Edward J Farrell et al, "Animated3D CT imaging", IEEE CG&A, Vol.5, No.12, Dec. 1985, pp. 26-32.

[9] Gerald Q Maguirej Jr., "Graphics Applied to Medical Image Registration", IEEE CG & A, March 1991 , pp 20-27.

[10] Marc Lovey, "A Hybrid Ray Traces for Rendering polygon and Volume Data", IEEE CG & A, Vol. 10, No.2, March 1990, pp 33-40.

[11] James F Greenleaf, "Computerized Tomography with Ultra Sound", Proce. of the IEEE, Vol. 71, No.3, March 1983, pp 330-337.

[12] Waldo S. Hinshaw et al, "An Introduction to NMR Imaging from the Bloch Equation to the Imaging equation", Procd. of the IEEE, March 1983, Vol 71, No.3, pp 338-350.

[13] SIEMENS, "Operating Instructions" SIRETOM2000, Computer Tomogram for head and neck, pp-6-9 & 14.

[14] Parvathi Dev, "Imaging and Visualization in Medical Education", IEEE CG&A, Vol.19, No.3, May-June 1999, pp 21-31.

[15] Steven Senger, "Visualizing and Segmenting large volumetric data sets", IEEE CG&A, Vol.19, No.3, May-June 1999, pp 32-37.

[16] Ludwing Adams et al, "Computer Assisted Surgery", IEEE CG&A, May 1990

[17] Linda J Brewster et al, “Interactive Surgical Planning”, IEEE CG&A, Vol.4, No.3, March 1984.

[18] Michael L Rhodes, “Computer Graphics in Medicine”, IEEE CG&A, Vol.10, No.2, March 1990.

[19] R.H.Taylorr, Stephane Lavallee et al, "Computer Integrated Surgery", Cambridge, MA:MIT Press 1995.

[20] G. Champleboux, Stephane Lavallee, et al "Accurate calibration of cameras and range imaging sensors, the NPBS method", IEEE Int'l conf. on Robotics and Automation", Nice France, May 1992, pp 1552-1558.

[21] P.Sautot, P.Cinquin, S. Lavallee, and J.Troccaz, "Computer Assisted spine surgery: A first step towards clinical application in orthopaedics",14th IEEE Eng. in Medicine & Biology conf. Paris, 1992, pp 1071-1072.

[22] Stephane Lavallee, Richard Szeliski, "Recovering the position and orientation of Free-Form objects from image contours using 3D distance maps", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.17, No.4, April 1995, pp 378-390.

[23] B.Parvin and G.Medioni, "B-Rep from Unregistered Multiple Range images", Proc. IEEE Int't conf. Research and Applications, May 1992, pp 602-607.

[24] G.Turk and M.Levoy, "Zippered polygon meshes and Shells", Proc. Computer Graphics SIGGRAPH-1994, pp 829-832.

[25] K.Higuchi, et al, "Building 3D models from Unregistered range images", CVGIP-GMIP, Vo.57, No.4, July 1995, pp 315-333.

[26] A.A.G.Requicha and J.R.Rossignac, "Solid Modeling and Beyond", IEEE CG & A, Vol.12, No.5, Sept 1992, pp 31-44.

[27] Jung Hyun Han and A.A.G.Requicha, "Feature Recognition from CAD Models", IEEE CG & A, Vol.18, No.2, March-April 1998, pp 80-94.

[28] D.D.Bedworth et al, "Computer integrated Design and Manufacturing", McGraw-Hill, New York, 1991.

[29] S.M.Goldwasser et al, "Physicians Work Station with real time performance", IEEE CG & A, Vol.5, No.9, 1985, pp 44-57.

[30] R.Lenz et al., "Presentation and perception of 3D images", Pictorial Information systems in Medicine, NATO ASI Series F: Computer and Systems Science, Vol. 19, Springer-Verlag, Berlin, 1986, pp 459-468.

[31] K.H. Hoehne et al, "Shading 3D Images from CT using Gray level Gradients", IEEE Trans. Medical Imaging, Vol. MI-5 , No.1, March 1986 , CS Press, Los Alamitos, Calif., pp 45-47.

[32] K.H. Hoehne et al, "Combined Surface display and reformatting for the 3D Analysis of Tomographic Data", Investigative Radiography, Vol.22, No.7, July 1987, pp 658-664.

[33] K.H. Hoehne et al, "Viewing Operations for 3D-Tomographic Gray Level Data", Computer Assisted Radiology (Proc. CAR), Springer-Verlag, Berlin, 1987, pp 599-609.

[34] W.K.Smith et al, "Algorithms for Ray Tracing Representations of Medical Image Data", Computer Assisted Radiology (Proc. CAR), Springer-Verlag, Berlin, 1987, pp 882-887.

[35] W.E. Loransen et al, "Marching Cubes: A High resolutions3D surface construction Algorithm", Computer Graphics (Proc.SIGGRAPH), Vol.21, No.3, July 1987, pp 163-169.

[36] R.A.Drebin et al, "Volume Rendering", Computer Graphics (Proc.SIGGRAPH), Vol.22, No.4, Aug 1988, pp 65-74.

[37] K.H. Hoehne et al, "3D Visualization of Tomographic Volume Data using the Generalized Voxel Model", Proc. Chapel Hill Workshop Volume Visualization, Univ. of North Carolina, Chapel· Hill,,N.C.,1989, pp 51-57.

[38] Cline H.E., Lorensen W.E., Kikinis R, Jolesz F, "Three Dimensional Segmentation of MR images of the head using probalility and connectivity", Journal of Computer Assisted Tomography, 14,6, 1990, pp 1037-1045.

[39] Paul E Summers, Abhir H. Bhalerao, David J. Hawkes, "Multiresolution model based Segmentation of MR Angiograms", Journal MRI, Nov-Dec 1997, Vol.7, No.6, pp 950-957.

[40] Klinger A.and Dayer C.R., "Experiments in picture representation using rectangular decomposition", Computer Graphics and Image Processing, March 1976, pp 68-105.

[41] A.Unnikrishnan, Y.U. Venkatesh and Priti Shankar, "Threaded Linear hierarchical quardtrees for computation of geometric properties of binary images", IEEE Trans on S\W Engg, Vol.14, No.5, May 1988 , pp 659-665.

[42] A.Unnikrishnan, Y.U. Venkatesh and Priti Shankar, "Connected Component Labeling using Quadtrees - A Bottom-up Approach", The computer journal,Vol.30, No.2, April 1987, pp 176-182.

[43] Chris Jackins and Steven L. Tanimoto, "Octrees and their use in representing three dimensional object", Computer Graphics and Image Processing, Vol.14, 1980, pp 249.

[44] Hunder G.M., "Efficient computation and data structures for graphics", Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, Princeton Uty, NJ, 1978.

[45] Reddy D.R., Rubin S., "Representation of three dimensional objects", CMU-CS-78-113, Computer Science Dept. Carnegie - Mellon University, Pittsburgh, April 1978, [Volumes] D.1.3 A.1.4

[46] Chen HH, Huang TS, "A Survey of Construction and Manipulation of Octrees", CVGIP, 1988, pp 409-431.

[47] Hunder G.M, Steiglitz, "Linear Transformations off pictures represented by quardtrees, IEEE CG & A, Vol.10 , No.3 , 1979 , pp 289-296.

[48] Meagher D, "Geometric Modelling using Octree encoding, Comp. Graphics Image Process, Vol.19,1982, pp129-147.

[49] Unnikrishnan B .S., Mercy Paul, Unnikrishnan A. , "Generation of Octrees of Solids from Revolution and Sweep" Informatics and communication CSI journal, Oct 1994.

[50] Mercy Paul, A. Unnikrishnan, "Octree encoding of solids" proceedings of ISTE Summer school on parallel processing Cochin University of Science and Technology, Cochin, Kerala, May, 1993.

[51] K. Yamaguchi et al, "Octree related data structures and algorithms", IEEE CG&A,, Vol 4, No.1, January 1984, pp53-59.

[52] Irene Gargantini, "Linear Octree for fast processing of three dimensional objects", IEEE Computer Graphics and Image Processing, 20, 4, December 1982, pp 365-374.

[53] Atkinson, Gargantini, Walsh, "Filling by quadrants or octants, CVGIP, Vol. 33, No. 2, 1986, pp138-155.

[54] I. Gargantini, M.V.S. Ramanath and T.R.S. Walsh "Linear octtrees: from data acquisition or creation to display", Proc of Computer Graphis-86, Vol.3, Anaheim, CA, pp 615-621, May 1986.

[55] Hanan Samet, and M. Tamminen, "Computing geometric properties of images represented by linear quardtrees", IEEE Trans. on Patern Analysis and Mc. Intl., Vol.7, No.2, pp 229-240, March 1985.

[56] C.A. Shaffer and Hanan Samet, "Optimal quardtree construction algorithms", Computer Vision, Graphics, and Image Processing, 37,3, pp 402-419, March 1987.

[57] Georges-Pieree Bonneau, "Multiresolution Analysis on Irregular Surface Meshes", IEEE Trans. on Visualization & Computer Graphics, Vol.4, No.4, Oct-December 1998.

58 Djaffer Ibaroudene and Raj Acharya, "Parallel Display of Objects Represented by Linear Octrees", IEEE Trans. on Parallel & Distributed Systems, Vol.6, No.1, January 1995.

59 Dirk Krechel, et al "Automatic Registration of MRI Head Volumes using an Octree Anatomical Atlas", 12th IEEE Symp. on Computer -Based Medical Systems, 1998.

60 Sojan Lal P., A Unnikrishnan, K. Poulose Jacob, "Parallel Implementation of Octree Generation Algorithm", IEEE Int. Conference on Image Processing ICIP, October 1998, Chicago, USA.

61 Pulli K., et al, "Robust meshes from multiple range maps", Proc. of the IEEE Int. Conference on Recent Advances in 3D Digital imaging and Modeling", 1997.

62 Jane Wilhems et al, "Octrees for faster isosurface Generation", ACM Trans. on Graphics, Vol 11, Issue 3, 1992.

63 Ned Greene et al, "Error bounded antialiased rendering of complex Environments", Proc. of 21st ACM Annual Conf. on Computer Graphics, July 1994, Orlando, FL, USA.

64 Stephane Lavallee et al, "Recovering the Position & Orientation of Free Form Objects from Image Contours Using 3D Distance Maps", IEEE Trans. on Pattern Analysis & Machine Intelligence Vo.17, No.4, April 1995.

65 Kyu-Young Whang, et al, "Octree-R: An Adaptive Octree for Efficient Ray Tracing", IEEE Trans. on Visualization and Computer Graphics, Vol. 1, No.4, December 1995.

66 Raghu Machiraju et al, "Structure-Significant Representation of Structured Datasets", IEEE Trans. on Visualization & Computer Graphis, Vol.4, No.2, April-June 1998.

67 Oded Sudarsky et al, "Dynamic Scene Occlusion Culling", IEEE Transactions on Visualization & Computer Graphics, Vol.5, No.1, January-March 1999.

68 Navazo I, "Geometric Modeling of octree encoded polyhedral objects", Doc. Thesis, Systems Informatics University *Plitecnica De Catalunya*, Barcelona, Spain 1986

69 Deborah Silver, Xin Wang, "Tracking and Visualizing Turbulent 3D Features", IEEE Trans. on Visualization and computer Graphics", Vol.3, No.2, April-June 1997, pp 129-141.

70 Hanan Samet, Reading of, "The Design and Analysis of Spatial Data Structures", Addison-Wesley 1989.

71 I. Gargantini, T.R..Walsh, and O.L. Wu, "Viewing transformations of voxel based objects via Linear octrees", IEEE CG & A, Vol.6, No.10, 1986.

72 D.Silver, et al "Ellipsoidal Quantification of Evolving phenomena", Visualization of Physical Phenomena", Proc.91 Int'l Symp. Computer Graphics, N.M.Patirkalakis, ed., pp 573-588. Springer-Verlag June 1991.

73 F.Post, et al, "Iconic Techniques for Feature Visualization", Proc. IEEE Visualization Oct-1995, pp 288-325.

74 M.Inui, M.Miura, F.Kimura, "Analysis of position Uncertainties of parts in an assembly using configuration space in octre representation", Solid Modeling 95-ACM, Salt Lake City, Utah USA, pp 73-81.

75 Sojan Lal P., Mercy Paul, A Unnikrishnan, K. Poulose Jacob, "Generation of Octree from CAT Scan Slices", Proce. of NET-X '97, National Seminar, Computer Society of India, May 1997, India.

76 Man-May Yau, "Generating quad trees of cross sections from Octrees", Computer Vision, Graphics and Image Processing, Vol.27, 1984, pp 211-238.

77 Hanan Samet, "An overview of Quadtrees, Octree and Related Hierarchical data structures", Proc. of the NATO advanced study institute on Theoretical Foundations of Computer Graphics and CAD held at IICiocco, Italy, July 1987,pp 4-17

78 Hanan Samet, "Quadtree and related Hierarchical Data structures", Computing surveys, vol.16, No.2, June 1984.

79 A Unnikrishnan, "Some hierarchical data structures based on Quadtrees", Ph.D dissertation, IISc., Bangalare, October 1987.

80 D.F.Watson, et al "Systematic Triangulations", Computer Vision, Graphics, and Image Processing 26,2, May 1984, pp 217,223.

81 A. Saalfeld, "Triangulated data structures for map merging and other applications in Geographic information systems", Proc. of the Int'l Symp. on Geographic Information Systems, Vol.3, Arlington, VA, Nov 1987, pp 3-13.

82 O.D. Faugeras, et al "Polyhedral approximation of 3D objects without holes", Computer Vision, Graphics, and image processing, 25, 2, Feb 1984, pp 169-183.

83 J.L.Posdamer, "Spatial sorting for sampled surface geometrics", Proc. of SPIE-Biostereometrics'82, 361, San Diego, August 1982,

84 Gideon Frieder et al, "Back-to-front display of voxel based objects", IEEE CG & A, Vol.5, No.1, Jan1985, pp52-60.

85 Jayaram K. Udupa and Dewey Odhner, "Fast Visualization, manipulation, and analysis of Binary Volumetric objects", IEEE CG & A, Vol., No., Nov.1991, pp 53-62.

86 Marc Levoy, "Volume rendering by adaptive refinement", Visual Computer,Vol.6, No.1, Jan.1990.

87 Marc Levoy,"Efficent Ray tracing of Volume data", ACM Trans. on Graphics, Vol.9, No.3, July 1990.

88 Henry Fuches, Marc Levoy, and Stephen M. Pizer, "Interactive Visualization o 3D medical data", COMPUTER, Vol.22, No.8, Aug 1989, pp 46-51.

89 E.Keppel, "Approximating Complex surfaces by triangulation of contour lines", IBM J. Research and Development, Vol.19, No.1, Jan 1975, pp 2-11.

90 H.Fuches et al, "Optimal Surface reconstruction from Planner contours", Comm. ACM, Vol.20, No.10, Oct. 1977, pp 693-702.

91 G.T.Hermann and H.K.Liu, "Three Dimensional Display of Human Organs from Computed Tomograms", Computer Graphics and Image Processing, Vol.9, 1979, pp 1-21.

92 E.K.Fishman et al, "Volumetric Rendering Techniques: Applications for 3 Dimensional imaging of the Hip", Radiology, Vol.163, 1987, pp 737-738.

93 W.W.Scott, E.K.Fishman and D.Magid,"Accetabular Fractures: Optimal Imaging", Radiology, Vol.163, 1987, pp 537-539.

94 J.E.Kuhlman et al., "Nonunion of Acetabular Fractures: Evaluation with interactive multiplanar CT", J.Orthopedic Trauma,,Vol.3, No.1, 1989, pp 33-40.

95 J.E.Kuhlman et al., "Complex Shoulder Trauma:3D CT Imaging", Orthopedics, Vol. 11, No.2, 1988, pp 1561-1563.

96 D.Magid et al., "Adult Ankle Fractures: Comparison of plain films and interactive two and three-dimensional CT scans", American J. Roentgenology, Vol.154, May 1990 pp 1017-1023.

97 D.N.Levin et al, "Surface of the Brain:3D MR images created with Volume rendering", Radiology, Vol.171, 1989, pp 277-280.

98 M. W.Varnnier et al, "3D CT reconstruction Images for craniofascial Surgical planning and evaluvation", Radiology, Vol.150, 1984, pp 179-184.

99 E.K.Fishman et al, "Advanced 3D revaluation of Accetabular Trauma: Volumetric Image processing", J.Trauma, Vol.29, No.2, 1989, pp 214-218.

100 Johathan Shade, Steven Gortler, LI-Wei, and Richard Szeliski, "Layered Depth Images", In Proceedings of SIGGRAPH- 1998, pp 231-242.

101 Dani Lischinski and Ari Rapppoport, "Image based rendering for non-diffuse synthetic scenes. Rendering Techniques '98, Proc. 9th Eurographics workshop on Rendering, 1998.

102 Brain Curles and Marc Levoy, "A Volumetric Method for Building Complex Models from Range Images", Proceedings of SIGGRAPH 1996, pp 303-312.

103 Ake Wallin, "Constructing Isosurface from CT Data", IEEE CG & A, Vol., No., Nov 1991, pp 28-33.

104 C.H. Chien, et al "Generation of Volume Surface Octree from Range Data", The Computer Society Conf on Computer Vision and Pattern Recognition, June 1988, pp 254-260.

105 C.I. Connoly, "Cumulative Generation of Octree Models from Range Data", Proceedings of Int'l Conf. Robotics, March 1984 pp 25-32.,

106 A.Li et al, "Octree Encoding of Objects from Range Images", Pattern Recognition, 27(5), May1994, pp 727-737.

107 Vijay Chandru, Swamy Manohar, E.D. Prakash, "Voxel-Based Modeling for Layered ManufactUring", IEEE CG & A, Vol.15, No.6, 1995pp 42-47.

108 V. Chandru, S. Manohar, "G-WoRP: A Geometric workbench for rapid prototyping", Proc. ASME Int'l Mechanical Engineering Congress, ASME, New York, 1994.

109 P.F. Jacobs, "Rapid prototyping and manufacturing - Fundamentals of stereo-lithography, Society of Manufacturing Engineers SME-CASA, Merborn, Mich. 1992.

110 M.Burns, "Automated Fabrication", PrentIce Hall, Englewood Cliffs, N.J., 1993.

111 Baining Guo, "A Multiscale Model for Structure Based Volume Rendering, IEEE Trans. on Visualizatio and Computer Graphics, Vol.1, No.4, 1995, pp 291-301.

112 H.Edelsbrunner, et al, "On the shape of a set of points in the plane", IEEE Trans. Information Theory, Vol.29, 1983, pp 551-559.

113 H.Edelsbrunner, et al, "Three-dimensional alpha shapes", ACM Trans. Graphics, Vol.13, 1994. pp 43-72.

114 N.Max et al, "Arean and Volume coherence for efficient visualization of 3D scalar functions". Computer Graphics, Vol.24, Nov. 1990, pp 27-33.

115 P.Shirley and A. Tuchman, "A Polygonal approximation to direct scalar volume rendering", Computer Graphics, Vo.24, Nov 1990, pp 63-70.

116 C.Stein, B. Becker, and N.Max, "Sorting and hardware assisted rendering for volume visualization", 1994 Symp. Volume Visualization, Washigton D.C, Oct 1994, pp 83-89.

117 Philip Slusallek, Hans Peter seidel, "Vision - An Architecture for Global Illumination Calculations", IEEE CG & A, Vol.1, No.1 March 1995, pp 77-96.

118 J.K.Udupa et al, "Surface and Volume rendering in 3D imaging: A Comparison", J. Digital Imaging, Vol.4, 1991, pp 159-168.

119 W.Lorenson and H.Clive, "Marching Cubes: A high resolution 3D surface construction algorithm", Computer Graphics, Vol.21, 1987, pp 163-169.

120 P. Nig and J.Bloomenthal, "An evaluation of implicit surface tilers", IEEE CG & A, Vol.13, Nov. 1993, pp 33-41.

121 Schroeder, W.J.Zarge, et al, "Decimation of triangle meshes", ACM Compter Graphics, No.26, 1992, pp 65-70.

122 Shu R, Chen Z, Kankanhalli M.S.,"Adaptive Marching Cubes", The Visual Computer , 11, 1995, pp 202-217.

123 Mountani et al, "Discretized Marching cubes", Proceedings of Visualization, 1994, pp 281-28'.

124 Wilhelms J, Van Gelder, "A Octree for faster isosurface generation", ACM Trans. Graphics,11, 1992, pp 201-227.

125 Raj Shekhar, Elias Fayyad, Roni Yagel, Fredrick Cornhill, "Octree-Based Decimation of Marching Cubes Surfaces", Sym on Volume Visualization, Oct 28-29, 1996, San Fransisco, CA, USA, pp 335-342.

126 Frederick M. Weinhaus, and Venkat Deverajan, "Texture Mapping 3D Models of Real-world Scenes", ACM Computing Surveys, Vo.29, No.4, December 1997, pp 325-365.

127 D.Laurr, P. Hanrahan, "Hierarchical Splatting:A progressive refinemcnt algorithm for Volume Rendering", Computer Graphics, Vol 25, July 1991, pp 285-288,

128 L.Westover, "Interactive Volume Rendering", Computer Graphics, Vol. 24, Aug 1990, pp 367-376.

129 Wilhelms J, Van Gelder, "A coherent projection approach for direct Volume Rendering", Computer Graphics, Vol.25, July 1991, pp 275-284.

130 P.Lacroute and M.Levoy, "Fast Volume rendering using a shear-wrap factorization of the viewing transformation", Computer Graphics, Vol.28, Ann. conf. series, 1994, pp 451-458.

[131] T.Totsuka and M.Levoy, "Frequency Domain Volume Rendering", Computer Graphics, Vol:27, Ann. Conf. Series 1993, pp 271-278.

[132] T. Malzbender, "Fourier Volume Rendering", ACM Trans. Graphics, Vol.12, July 1993.

[133] Bomjun Kwon et al, "Memory Efficient Ray Classification to Visibility Operations", IEEE Trans. on Visualization and Comp. Graphics, July -Sept. 1998, pp 193-201.

[134] J Arvo and D Kirk, "A survey of Ray tracing acceleration techniques", An introduction to Ray Tracing, A.S. Glassner ed, Academic Press, 1989.

[135] J D. Foley et al, Computer Graphics: Principles and Practices, Second ed, Addison Wesley, 1990.

[136] E Rainhard, A.G. Chalmers and F W. Jansen, "Overview of Parallel photo-realistic Graphics" Proc. of Eurograhics'98, 1998.

[137] M.Lovey, "Display of Surfaces from Volume Data", IEEE CG&A, Vol.8, No.3, 1988, pp 29-37.

[138] A. Kaufman, Volume Visualization, IEEE CS Press, 1991

[139] L. Sobierajski and A Kaufman, "Volumetric Ray Tracing", Proc. of Workshop on Volume Visualization, pp 11-18, 1994.

[140] T. Whittened, "An Improved Illumination Model fo Shaded Display", Comm. CM Vol.23, June 1980, pp 343-349.

[141] Hans Peter Meinzer et al "Th Heidelberg Ray Tracing Model", IEEE CG & A, Vol., No., Nov 1991, pp 35-43.

[142] K.L. Ma, J.S.Painter, C.D. Hansen and M.F. Krogh, "Parallel Volume Rendering Using Binary Swap Composing", IEEE CG & A, Vol. 14, No.4, July 1993, pp 59-68.

[143] M.J.Muuss, "RT REMRT - Shared Memory Parallel and Network Distributed Ray Tracing Programs" USENIX: Proc. Forth Computer Graphics Workshop, Oct 1987.

[144] G.Vezina, P.A.Fletcher and P.K.Robertson, "Volume Rendering on the MasPar MP-1", Proc. 1992 Workshop Volume Visualization, Boston, 19-20, Oct 1992, pp 3-8,

[145] P.Schroder and G Stoll, "Data Parallel Volume Rendering as line drawing", Proc. 1992 Workshop Volume Visualization, Boston, 19-20, Oct 1992, pp 25-31.

[146] M.J.Muuss, "Towards Real Time Ray Tracing of Combinational Solid Geometric Models", Proc. BRL-CAD Symp. June 1995.

147 S.Whitman, "A Survey of Parallel Algorithms for Graphics and Visualization, "Proc. High Performance Computing for Computer Graphics and Visualization, Swansea, 3-4 July 1995, pp 3-22.

148 Steven Parker et al, "Interactive Ray Tracing for Volume Visualization", IEEE Trans on Visualization and Computer Graphics, Vol. 5, No.3, July-Sept 1999, pp 238-250.

149 Heung-Yeung Shum et al, "An Integral approach to Free-Form object modeling", IEEE Trans. on Patern Analysis and Machine Intelligence, Vol.19, No.12, Dec 1997, pp 1366-1370.

150 Derek R. Ney and Elliot K. Fishman, "Editing Tools for 3D Medical Imaging", IEEE CG & A, Vol., No., Nov 1991, pp 63-71.

151 Thomas W. Ryan, L. Darwin Sanders, H.Donald Fisher, and A. Evan Iverson, "Image compression by Texture Modeling in the Wavelet Domain", IEEE Trans. on Image Processing, Vol. 5, No.1, January 1996.

152 N.M. Nasrabadi and R.A. King, "Image coding using vector quantization: A review", IEEE Trans. Commun., Vol.36, pp 957-971, Aug 1988.

153 D. cohn, E.A. Riskin, and R. Ladner, "Theory and Practice of vector quantizers trained on small training set", IEEE Trans. Pattern Anl. Machine Intell. Vol. 16, pp 54-65, Jan 1994.

154 Weiping Li, Ya-Qin Zhang, "Vector Based Signal processing and Quantization for Image and Video compression", Proceedings of the IEEE, Vo. 83, No.2, February 1995.

155 Dong Sik Kim, Taejeong Kim, Sang U.K, "On testing Trained Vector Quantizer Codebook", IEEE Trans. on Imge Processing, Vol. 6, No.3, 1997.

156 Eric J. Stollnitz, "Wavelets for computer Graphics: A Primer Part 2", IEEE CG&A, Vo.15, No.4, July 1995.

157 Georges-Pierre Bonneau, "An Introduction to Wavelets for Scientific Visualization", Proc. of the IEEE Dagstuhl-97, Scientific Visualization conference.

158 S.Muraki, "Approximation and rendering of volume data using wavelet transforms", IEEE CG & A, Vol.13, No., 1993, pp 50-56.

159 R.Westermann, "A Multiresolution frame work for Volume Rendering", 1994 Symp. Volume Visualization, Washington DC, Oct 1994, pp 51-57.

160 S.Muraki, "Multiscale 3D edge representation of volume data by DOG wavelet", 1994 Symp. Volume VIsualizatIon,,Washington DC, Oct 1994, pp 35-42.

61 Georges-Pierre Bonneau, "Multiresolution Analysis on regular Surface Meshes", IEEE Trans. on Visualization and Computer Graphics", 1998, pp 365-378.

62 J.Hutchinson, "Fractals and self-similarity", Indiana Univ. Mathematics J., Vo.30, No.5, 1981, pp 713-747.

63 M.F.Barnsley and S.G.Demko, "Iterated Functions Schemes and the Global Construction of Fractals", Proc. Royal Soc. A, Vol.399, 1985, pp 243-275.

64 M.F.Barnsley et al, "Recurrent Iterated Function Systems", Constructive Approximation, Vo.5, 1989, pp 3-31.

65 G.Vines, "Signal Modeling with Iterated FunctIon SystEms", PhD thesis, Georgia Inst of Tehnology, May 1993.

166 A.E. Jacquin, "Image coding based on Fractal Theory of Iterated contractive image transformations", IEEE Trans. on Image Processing, vol. 1, No.1, Jan 1992, pp 18-30.

167 Y.Fisher, et al, "Fractal Image Compression using iterated transforms", Image and Text copression J.A.Storer, ed Boston:Kluwerr, 1992.

168 Waye O. Cochran et al, "Fractal Volume Compression", IEEE Trans. on Visualization and computer Graphics, vol.2, No.4, Dec 1996, pp 313-322.

169 Kwon Kim and Rae-Hong Park, "Still Image Coding based on Vector Quantization and Fractal Approximation", IEEE Trans. on Image Processing, Vol.5, No.4, April 1996, pp 587-597.

170 Sojan Lal P., A Unnikrishnan, K. Poulose Jacob, " Generation of Octrees from raster scan with reduced information loss", Communicated

171 S.K. Sinha, R.Krishna Kumar, and M.K.Champaka, "Transputers and Occam", Indian Institute Science, Bangalore, India, Summer School-Feb 7-11, 1994.

172 R.Dettmer, "Chip architectures for parallel processing", Electronics and Power,Vo.31, No.3,1985, pp 227-231.

173 A Gottlieb et al, "The NYU Ultra-computer: Designing a MIMD shared memory parallel computer",, IEEE Trans. on Computers, Vol.32, No.2, pp 175-189, Feb. 1983.

174 S.Chang, "A model for Distributed Computer System Design", IEEE Trans. Systems, Man, and Cybernetics, Vol.5, No.6, pp 344-359, 1979.

75 D.M. Pase, and A.R. Larrabee, "Intel iPSC concurrent Computer", Programming Parallel Processors, R.G. Badd, and R. Robert eds., pp 93-104, Addision-Wesley, 1988.

176 Mahamed Ada, "A Scalable Multibus Configuaration for Connecting Transputer Linkes", IEEE Trans. on Parallel and Distributed Systems, Vol.8, No.3, pp 245-253, March 1997.

177 INMOS, The Transputer Data Book, Second Edition, 1989.

178 G.W.Irwin, P.J.Fleming, Reading "TRANSPUTERS in REAL-TIME CONTROL", Research studies press LTD., England, and John Wiley and Sons Inc.

179 J.W.Ponton et al, "Non linear process simulation and control using transputers", IEEE Proc. D, 137,4, 1990, pp 189-196.

180 Fried Stephen, "Shared memory and PC super computing; achieving giga flop performance on a PC", Dr.Dobbs Journal, January 1994.

181 Warren Day, "Farming: towards a rigorous definition and efficient Transputer implementation", Proceedings of the 15th World Occam and Transputer user Group technical meeting 13-15 April,1992, Aberdeen, Scotland, UK.

182 Geoff Barrett et al, "Formal Methods in the Design of the T9000", Proceedings of the 15th World Occam and Transputer User group Technical Meeting, 13-15 April 1992, Aberdeen, Scotland, UK.

183 Robert M. Gary, "Vector Quantization", IEEE ASP Magazine, April 1984, pp 4-29.

184 Markus H Gross, Oliver G Staadt, Roger Gatti, "Efficient triangular surface approximation using wavelet and quardtree data structure", IEEE Transactions on Visualization and Computer Graphics, Vol.2, No.2, June 1996, pp 130.

185 Amir Avervuch, Danny Lazer, Moshe Israeli, "Image Compression using wavelet transform and multi-resolution decomposition", IEEE Transaction on Image Processing, Vol.5, No.1, Jan 1996, pp4.

186 A. Gersho and R.M. Gray, "vector quantization and signal compression" Boston: Kluwer, 1992.

187 Y. Linde, A Buzo, and R.M.Gray, "An algorithm for vector quantizer design", IEEE Trans. Commun. Vol. COM-28, pp 84-95, Jan 1980.

188 D.L. Neuhoff and R.G. Munoz, "Robust source coding of weakly compact classes", IEEE Trans. Inform. Theory, Vol IT-33, pp 522-529, July 1987.

189 R.M. Gray, and L.D. Davisson, "Quantizer mismatch", IEEE Trans. Commun. Vol. COM-23, pp 439-443, April 1975.

190 W. Mauersberger, "Experimental results on the performance of mismatched quantizers", IEEE Trans. Inform. Theory, Vol IT-25, pp 381-386, July 1979.

191 Y. Yamada, S. Tazaki, et al "Variance mismatch of vector quantizers", IEEE Trans. Inform. Theory Vo. IT-30, pp 104-107, Jan. 1984.

192 Dong S Kim, T. Kim, and S.U. Lee, "On testing trained vector quantizer codebook", IEEE Trans. on image Processing, Vol. 6, No.3, March 1997.

193 E.J.Stollnitz, T.D. DeRose, and David H. Salesin, "Wavelets for Computer Graphics:A Primer, Par 1", IEEE CG & A, Vol.15, No.3, 1995, pp 76-84.

194 E.J.Stollnitz, T.D. DeRose, and David H. Salesin, "Wavelets for Computer Graphics:A Primer, Par 2", IEEE CG & A, Vol.15, No.4, 1995, pp 75-85.

195 I. Daubechies, "Orthonormal bases of compactly supported wavelets", Comm. on Pure and Applied Mathematics", Vol 41, 1988, pp 909-996.

196 S.Mallat, "A Theory of Multiresolution Signal Decomposition: The wavelet representation", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol 11, No.7, July 1989, pp 674-693.

197 D. Berman et al, "Multiresolution Painting and Composing", Siggraph 94, Conf. Proc. ACM, New York, 1995.

198 R. DeVore et al, "Image Compression Through wavelet Transform Coding", IEEE Trans. information Theory, Vol 38, No.2, March 1992, pp 719-746.

199 C.E.Jacobs et al, "Fast Multiresolution Image Querying", Siggraph '95, ACM, New York, 1995.

200 A. Finkelstein et al, "Multiresolution Curves", Siggraph 94, Conf. Proc. ACM, New York, 1994, pp 261-268.

201 S.G. Gortler and M.F.Cohen, "Hierarchical and Variational Geometric Modeling with Wavelets", Proc. Symp. on Interactive 3D Graphics, ACM, New York, 1995.

202 M. Lonsbery et al, "Multiresolution Surface of Arbitrary Topological Type", Tech Report 93-10-05B, Univ. of Wash., Dept. of CS and Eng., Seattle, Wash, 1993.

203 D.Mayers, "Multiresolution Tiling", Computer Graphics forum, Vo.13, No.5, Dec 1994, pp 325-340.

204 Z.Liu, S.J Gortler, M.F. Cohen, "Hierarchical Space time control", Siggraph 94, Conf. Proc. ACM, New York, 1994, pp 35-42.

205 P.H.Christensen et al, "Clustering for Glossy Global Illumination", Tech Report 95-01-07, University of Washington, Dept. of CS and Engg., ., Seattle, Wash, 1995

206 P.H.Christensen et al, "Wavelet Radiance", Proc. of fifth Eurographics workshop on rendering, held in Darmastadt, Germany, 1994, pp 287-302.

207 S.J. Gortler et al, "Wavelet Radiosity", Siggraph 93, Conf. Proc. ACM, New York, 1993, pp 221-230.

208 P. Schroder et al, "Wavelet Projections for Radiosity", Proc. Fourth Eurographics workshop on rendering, held in Paris, 1993, pp 105-114.

209 G. Beylkin, R. Coifman, and V.Rokhlin, "Fast wavelet transforms and numerical algorithms", Communications of Pure and Applied Mathematics, 44(2); pp 141-183, March 1991.

210 Eric J. Stollnitz, Tony D. Derose, and David H Salesin, "Wavelets for computer Graphics, theory and applications", Morgan Kaufmann publishers, inc., San Fransisco, 1996, ISBN:1-55860-375-1.

211 Thomas W Ryan, Darurin Sanders, H. Donald Fisher, A. Evan Iversor, "Image compression by texture modeling in the wavelet domain", IEEE Transactions on Image Processing, Vol.5, No.1, Jan 1996, pp 26.

212 S. Muraki, "Volume data and wavelet transform", IEEE Computer Graphics & Applications, Vol. 13, pp 50-56, July 1993.

213 H. Fuchs, Z.M. Kedem, Uselton S.P., "Optimal Surface reconstruction fro- Planner contours", CACM, Vol. 20, No.10., pp 693-702, 1977.

214 Geoff Wyvill, Craig Mc Pheeters, and Brain Wyvil, "Data Structure for soft objects", The Visual Computer, 2: pp 227-234, 1986.

215 Schroeder, W.J. Zarge, J.A. Lorensen, "Decimation of triangle meshes", Computer Graphics, Vol.26, 1992, pp 65-70.

216 Raj Shekhar, Elias Fayyad, Roni Yagel, and Fredrick Cornhill, "Octree-Based Decimation of Marching Cubes Surface", Proceeding of Volume Visualization, pp 335-342, IEEE Computer Society Press, 1996..

217 Shu R, Chen Z, Kankanhalli M.S, "Adaptive Marching Cubes, The Visual Computer, Vol.11, pp 202-217, 1995

218 Montani C, Scateni R, Scopigno R, "Discretized Marching Cubes", Proceedings of Visualization, pp 281-287, 1994.

219 Wilhelms J., Van Gelder, "Octree for faster isosurface generation", Computer Graphics, Vol.24, No.5, pp 57-62, November 1990.

220 Wilhelms J., Van Gelder, "Octree for faster isosurface generation", ACM Trans. Graph., Vol.11, pp 201-227, 1992.

221 P. Cignoni, C.Montani, E.Puppo and R.Scopingo "Optimal Isosurface extraction from irregular volume data", Proc. of IEEE 1996 Symposium on Volume Visualization. ACM Press, 1996.

222 Y. Livnatt, H. Shen, and C.R. Johnson, "A near Optimal isosurface extraction algorithm using the span space". IEEE Trans. Vis. Comp. Graphics, Vol.2, No.1, pp 73-84, 1996.

223 Kwang-Man Oh and Kyu Ho Parkk, "A type-merging algorithm for extracting an isosurface from volumetric data", The Visual Computer, 12: pp 406-419, 1996.

224 Tim Poston, H.T. Nguyen, Pheng-An Heng and Tien-Tsin Wong, "Skeleton climbing: fast isosurfaces with fewer triangles", In Pacific Grahics 97, pp 117-126, Korea, October 1997.

225 Steven Parker, Peter Shirley, Yarden Livnnat, Chuck Hansen, and Peter Pike Sloan, "Interactive Ray Tracing for isosurface rendering" in Visualization'98, 1998.

226 Yarden Livnat, and Charles Hasen, "View Dependent Isosurface Extraction", Proce. of the IEEE conference on Visualization-98, pp 175-180, 1998,

227 Djaffer Ibaroudence and Raj Acharya, "Parallel Display of Objects represented by Linear Octrees", IEEE Trans. on Parallel and Distributed systems, Vo.6, No.1, Jan 1995